Programmazione

– Calcolatori e programmi –

Francesco Tiezzi

Scuola di Scienze e Tecnologie

Sezione di Informatica

Università di Camerino

Lucidi originali di Pierluigi Crescenzi

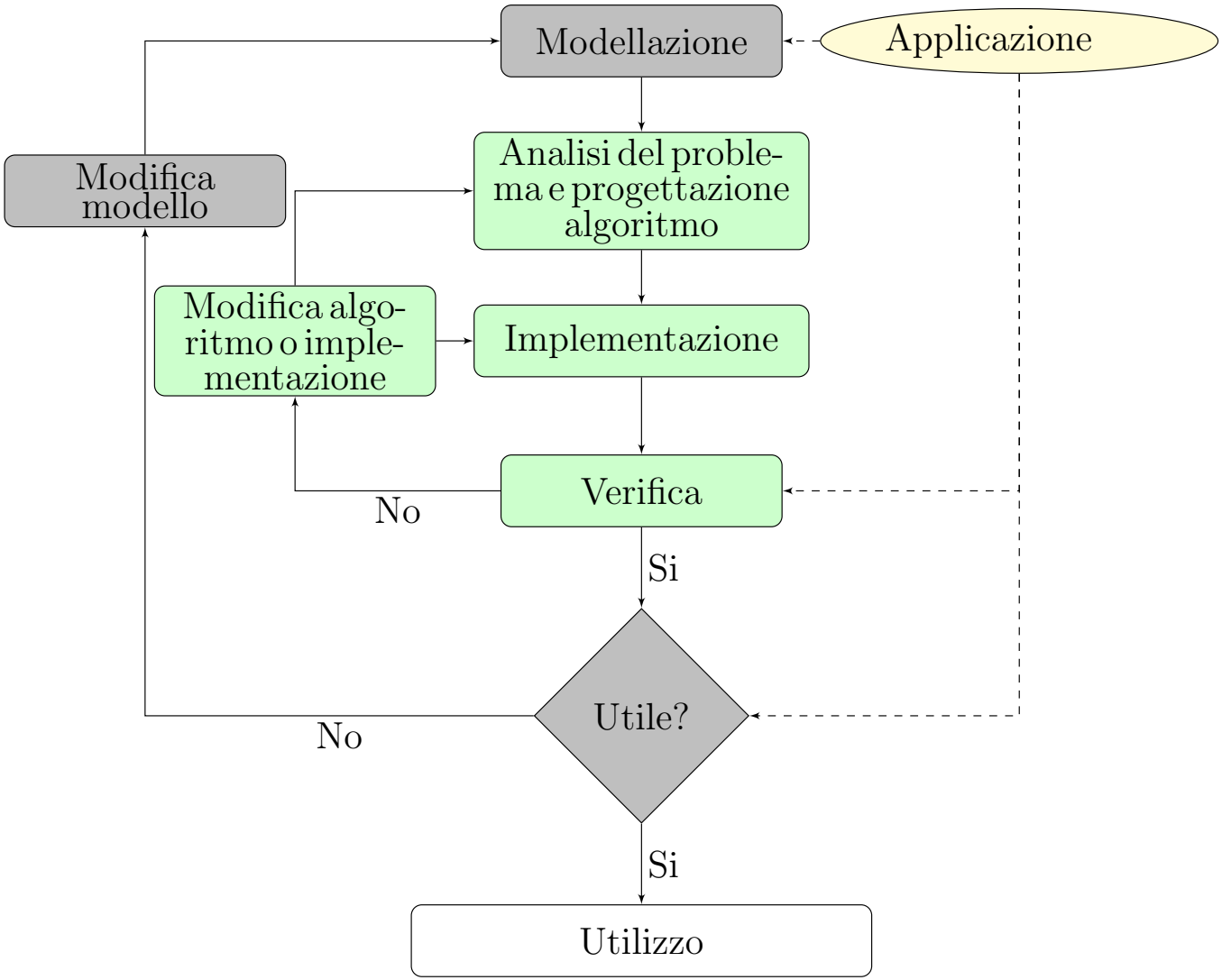
Cosa è l'informatica?

- Più facile dire cosa non è
  - Poco a vedere con "alfabetizzazione informatica" (saper usare un computer per scrivere un testo o navigare in Internet)
  - Non consiste semplicemente nello scrivere programmi

- Denning et al (1989)
  - *L'informatica è lo studio sistematico dei processi algoritmici che descrivono e trasformano l'informazione: la loro teoria, analisi, progettazione, efficienza, implementazione e applicazione*

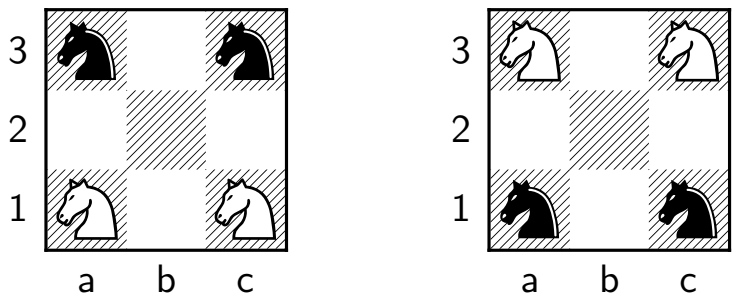
- Metodo algoritmico
  - Formulare algoritmi che risolvano un problema
  - Trasformare questi algoritmi in programmi
  - Verificare la correttezza e l'efficacia di tali programmi analizzandoli ed eseguendoli

Il metodo algoritmico

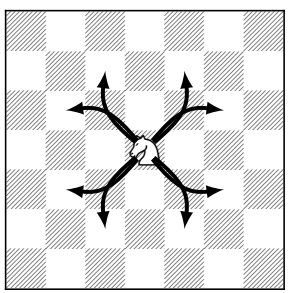


Il metodo algoritmico: trovare il giusto modello

- Qual è la sequenza di mosse più breve che consente ai cavalli di passare dalla configurazione a sinistra a quella a destra?

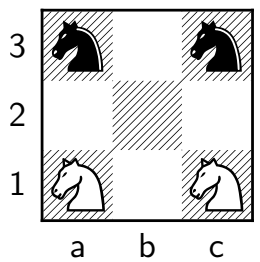


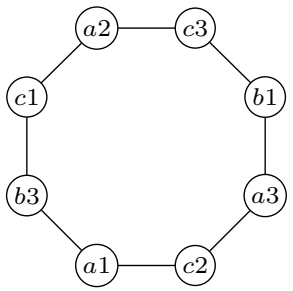
- Possibili mosse del cavallo



Il modello

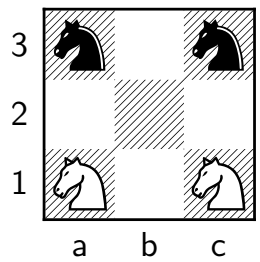
- Rappresentare il problema mediante una relazione di raggiungibilità

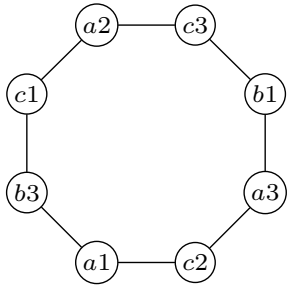




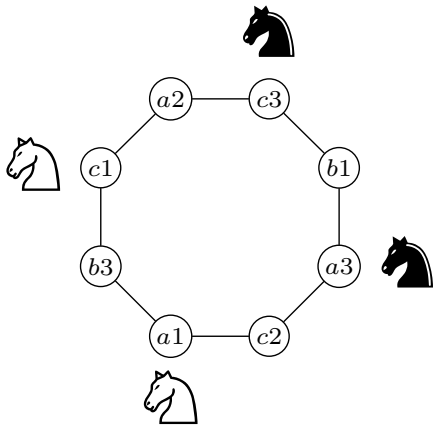
Il modello

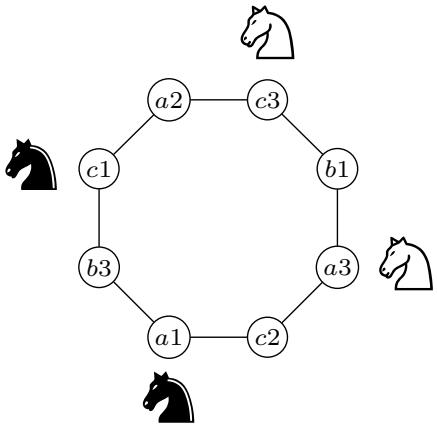
- Rappresentare il problema mediante una relazione di raggiungibilità





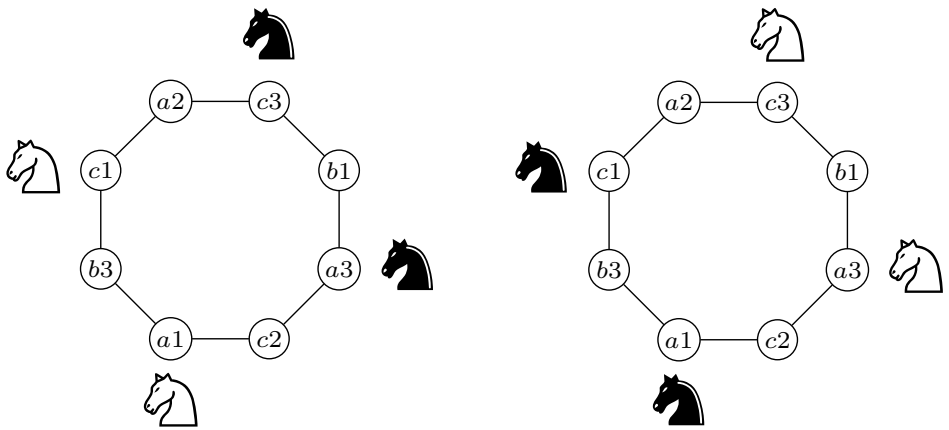
- Il problema diventa: trovare il minimo numero di mosse per andare dalla configurazione a sinistra a quella a destra





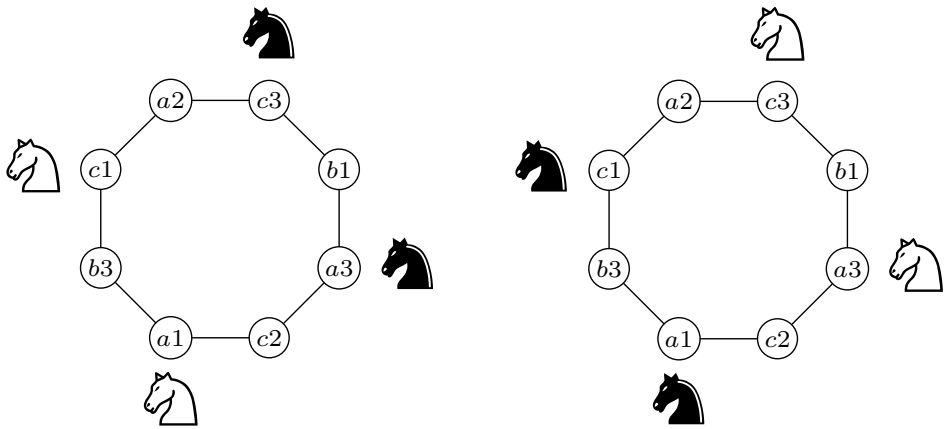
La soluzione: algoritmo

- Trovare il minimo numero di mosse per andare dalla configurazione a sinistra a quella a destra



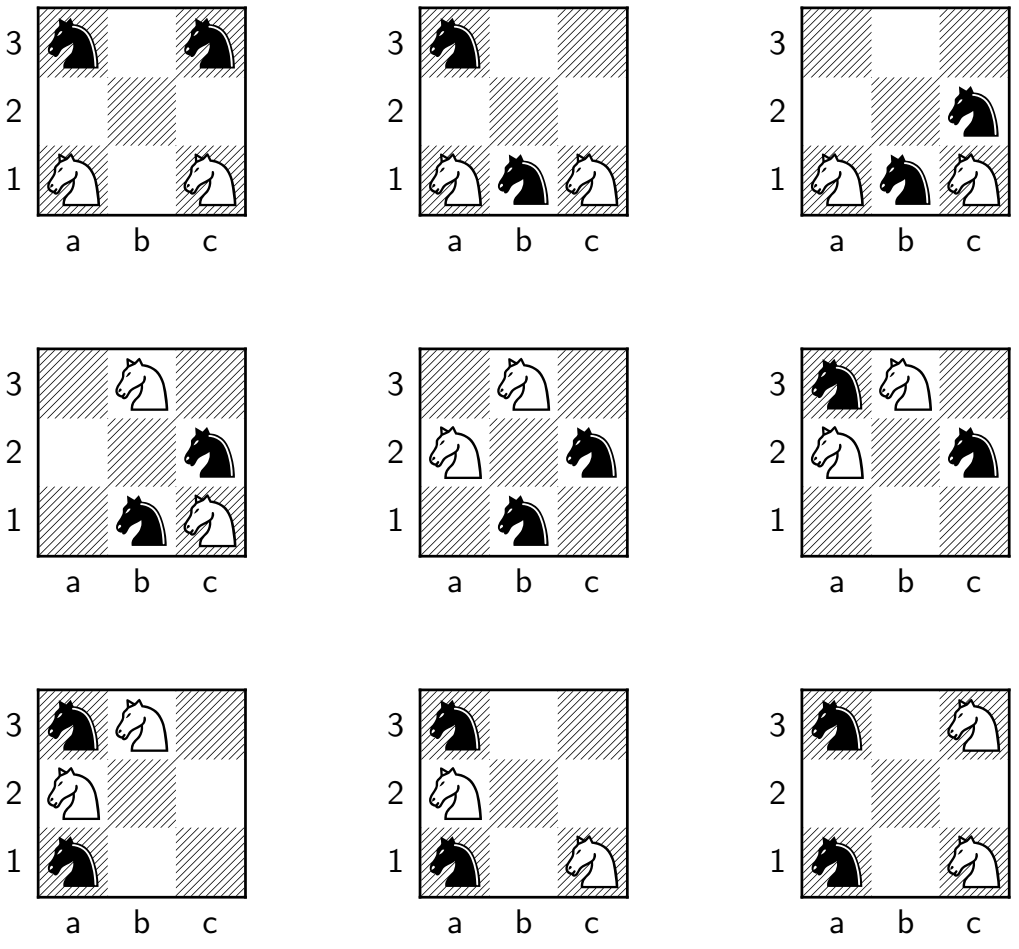
La soluzione: algoritmo

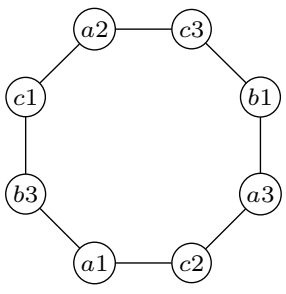
- Trovare il minimo numero di mosse per andare dalla configurazione a sinistra a quella a destra



- Ruotare i cavalli di quattro posizioni in senso orario (o antiorario)

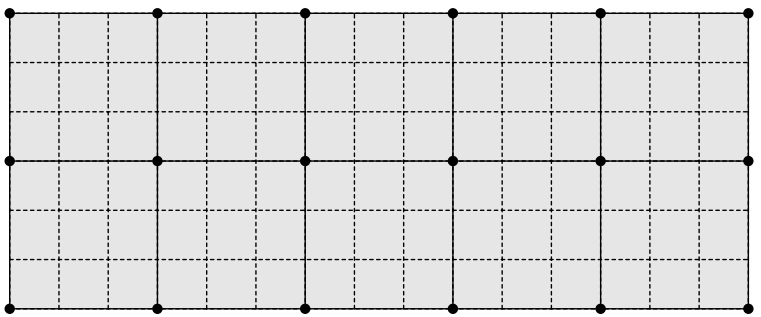
La soluzione: le prime 8 mosse





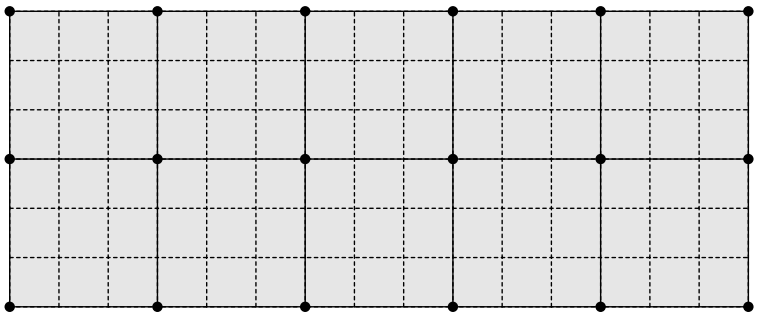
Il metodo algoritmico: trovare il giusto algoritmo

- Problema
  - Piastrellare una stanza rettangolare di dimensione $n \times m$ con il minor numero possibile di mattonelle quadrate di uguale dimensione
    - Esempio: $n = 6$ e $m = 15$



Il metodo algoritmico: trovare il giusto algoritmo

- Problema
  - Piastrellare una stanza rettangolare di dimensione $n \times m$ con il minor numero possibile di mattonelle quadrate di uguale dimensione
    - Esempio: $n = 6$ e $m = 15$



- Modello
  - Determinare il massimo numero intero che divide sia n che m
  - Calcolare il *massimo comun divisore* (MCD) di n e m
    - $MCD(6, 15) = 3$

Il metodo algoritmico: trovare il giusto algoritmo

Esistono diversi algoritmi per il calcolo del MCD di n e m

Il metodo algoritmico: trovare il giusto algoritmo

Esistono diversi algoritmi per il calcolo del MCD di n e m

- Algoritmo naïf:
  - supponiamo $n < m$ e n non divide m
  - esaminiamo tutti i numeri d tra $n/2$ e 2 (in ordine inverso): se d divide n e divide m, allora $MCD(n, m) = d$

Il metodo algoritmico: trovare il giusto algoritmo

Esistono diversi algoritmi per il calcolo del MCD di n e m

- Algoritmo naïf:
  - supponiamo $n < m$ e n non divide m
  - esaminiamo tutti i numeri d tra $n/2$ e 2 (in ordine inverso): se d divide n e divide m, allora $MCD(n, m) = d$

Inefficiente: nel caso pessimo ($MCD(n, m) = 1$) bisogna provare tutti i numeri tra $n/2$ e 2

Il metodo algoritmico: trovare il giusto algoritmo

Esistono diversi algoritmi per il calcolo del MCD di n e m

- Algoritmo naïf:
  - supponiamo $n < m$ e n non divide m
  - esaminiamo tutti i numeri d tra $n/2$ e 2 (in ordine inverso): se d divide n e divide m, allora $MCD(n, m) = d$

  Inefficiente: nel caso pessimo ($MCD(n, m) = 1$) bisogna provare tutti i numeri tra $n/2$ e 2

- Algoritmo di Euclide:
  - fintanto che $n \neq 0$ o $m \neq 0$, se $n < m$ passa alla coppia $(n, m \mod n)$, altrimenti passa alla coppia $(m, n \mod m)$

  Efficienza ottimale

Algoritmi

- **Informatica**: studio sistematico dei processi algoritmici che descrivono e trasformano l'informazione: la loro teoria, analisi, progettazione, efficienza, implementazione e applicazione

- **Algoritmo**: successione finita di istruzioni o passi che definiscono le operazioni da eseguire su dei dati (che formano l'istanza di un problema) per ottenere dei risultati (intesi come la soluzione dell'istanza specificata)
  - Proprietà
    - Finito
    - Generale
    - Non ambiguo
    - Corretto
    - Efficiente

Problemi indecidibili

- Non tutti i problemi (computazionali) ammettono algoritmi di risoluzione: **problema della fermata** (Turing, 1937)

Dato un generico algoritmo (o programma) A e dato un input x, A con x in ingresso **termina** o **va in ciclo**?

Problemi indecidibili

- Non tutti i problemi (computazionali) ammettono algoritmi di risoluzione: **problema della fermata** (Turing, 1937)

Dato un generico algoritmo (o programma) A e dato un input x, A con x in ingresso **termina** o **va in ciclo**?

- Esempio: algoritmo `P`
  - Con input n
    1. Pone $f = 2$
    2. Fintanto che $f < n$ e $n \mod f$ è diverso da 0, aumenta f di 1
    3. Se $f = n$, allora il numero è primo, altrimenti non lo è
- Per un numero n qualsiasi, `P(n)` termina?

Problemi indecidibili

- Non tutti i problemi (computazionali) ammettono algoritmi di risoluzione: **problema della fermata** (Turing, 1937)

Dato un generico algoritmo (o programma) A e dato un input x, A con x in ingresso **termina** o **va in ciclo**?

- Esempio: algoritmo P
  - Con input n
    1. Pone $f = 2$
    2. Fintanto che $f < n$ e $n \mod f$ è diverso da 0, aumenta f di 1
    3. Se $f = n$, allora il numero è primo, altrimenti non lo è
- Per un numero n qualsiasi, P(n) termina?
  - Si, perché f è aumentato di 1 a ogni ripetizione del passo 2 e a un certo punto deve divenire uguale a n

Problema della fermata

- Un algoritmo è una sequenza di simboli
  - Un algoritmo può essere dato in pasto a un altro algoritmo

Problema della fermata

- Un algoritmo è una sequenza di simboli
  - Un algoritmo può essere dato in pasto a un altro algoritmo
- Esiste un algoritmo $\mathtt{T}(A, x)$ che, in tempo finito, risponde SI se $A(x)$ termina, risponde NO se va in ciclo?

Problema della fermata

- Un algoritmo è una sequenza di simboli
  - Un algoritmo può essere dato in pasto a un altro algoritmo

- Esiste un algoritmo $\mathtt{T}(A, x)$ che, in tempo finito, risponde SI se $A(x)$ termina, risponde NO se va in ciclo?

  No

Problema della fermata

- Un algoritmo è una sequenza di simboli
  - Un algoritmo può essere dato in pasto a un altro algoritmo
- Esiste un algoritmo $\mathtt{T}(A, x)$ che, in tempo finito, risponde SI se $A(x)$ termina, risponde NO se va in ciclo?

  No
- Il problema della fermata è *indecidibile*
- Altri problemi lo sono: stabilire equivalenza tra 2 programmi

Esponenziale vs polinomiale

Consideriamo di poter eseguire 2 miliardi di operazioni al sec.

Esponenziale vs polinomiale

Consideriamo di poter eseguire 2 miliardi di operazioni al sec.

- Esponenziale

n	10	20	30	40	50	60	70
tempo: 2^n	512ns	524μs	537ms	9m	7g	18a	187s

Esponenziale vs polinomiale

Consideriamo di poter eseguire 2 miliardi di operazioni al sec.

- Esponenziale

n	10	20	30	40	50	60	70
tempo: 2^n	512ns	524μs	537ms	9m	7g	18a	187s

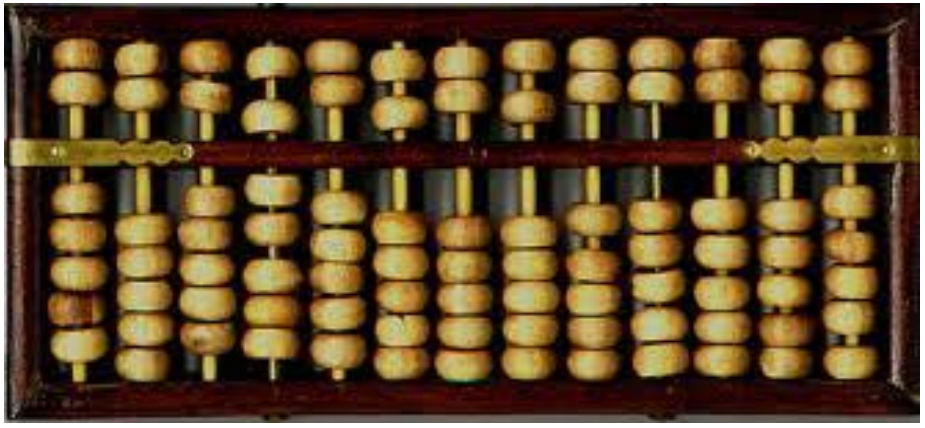
- Quadratico

n	10	20	30	40	50	60	70
tempo: n^2	50ns	200ns	450ns	800ns	1μs	2μs	3μs

Nozioni di base

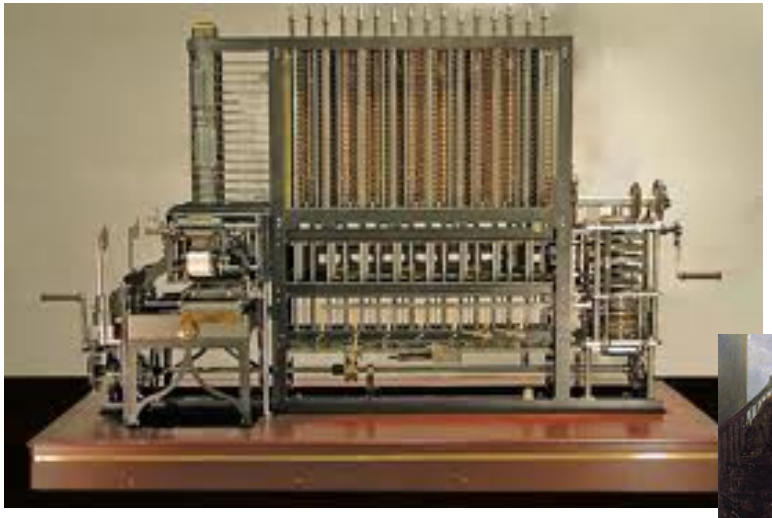
- Un calcolatore consiste di hardware e di software
  - **Hardware**: unità di elaborazione centrale, memoria principale, memoria ausiliaria, periferiche
  - **Software**: istruzioni raccolte in programma

Primi strumenti di calcolo

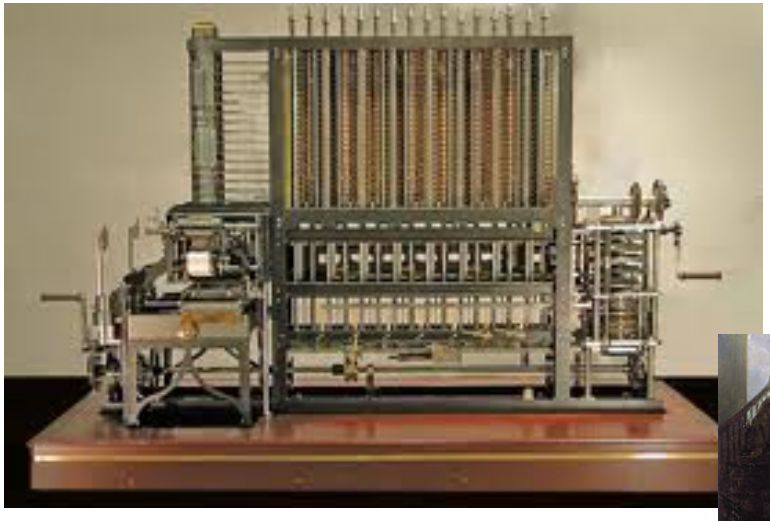


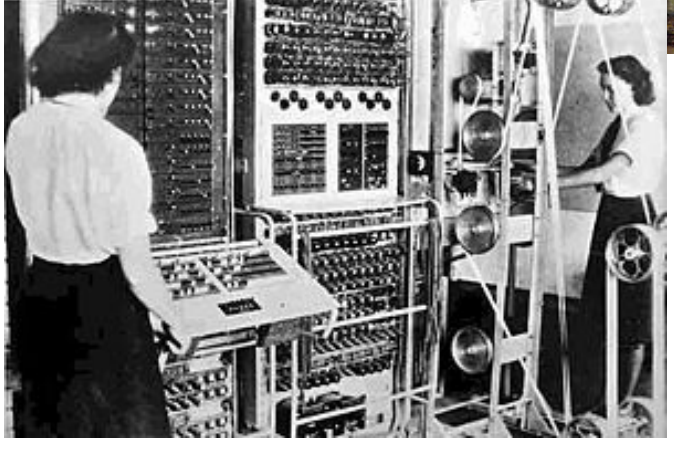
Primi strumenti di calcolo

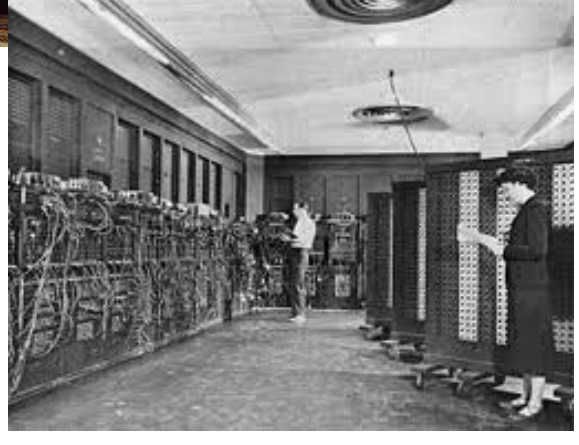
Primi calcolatori



Primi calcolatori

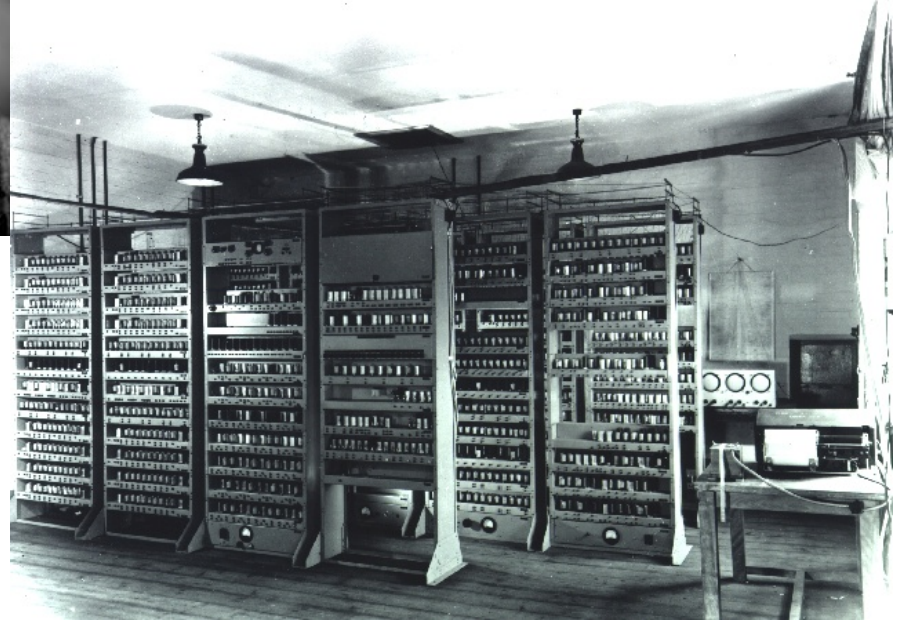




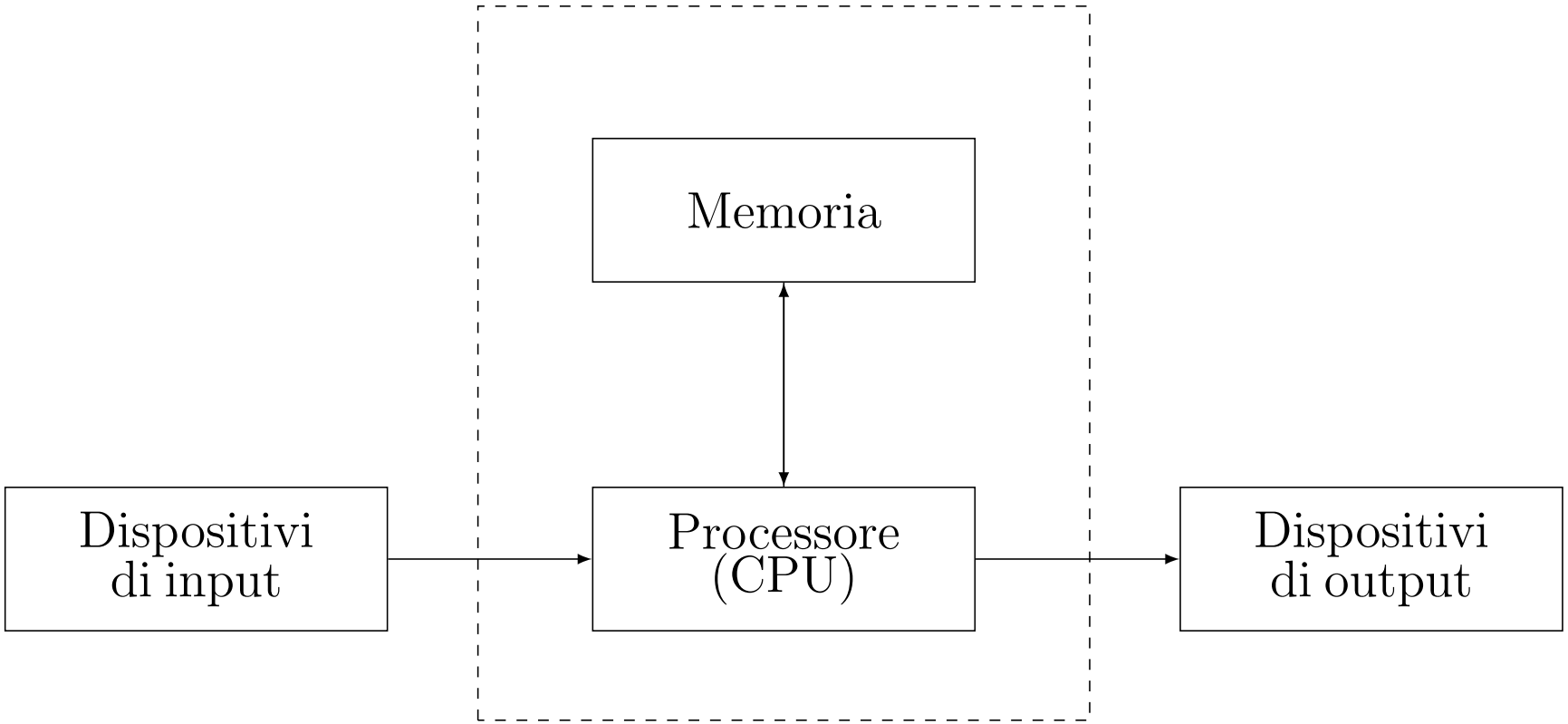


La macchina di Von Neumann

La macchina di Von Neumann



Componenti hardware principali



Componenti hardware principali

- **CPU**: dispositivo che esegue le istruzioni di un programma
  - Solo operazioni molto semplici, come trasferimento di un dato oppure operazioni aritmetiche elementari
- **Memoria principale**: veloce, ma costosa e volatile (conserva il programma attualmente in esecuzione ed i dati da esso usati)
- **Memoria ausiliaria**: meno costosa e che perdura anche in assenza di elettricità, ma più lenta (utilizzata per conservare programmi e dati in modo più o meno permanente)

- **Bit**: può assumere due soli valori (0 ed 1)
- **Byte**: pari a 8 bit (2^8 possibili valori)
- **Locazione di memoria**: sequenza di byte adiacenti associata al dato il cui indirizzo è l'indirizzo del primo byte di sequenza

Indirizzo	Dato	
...	...	...
484	00011110	Primo dato:
485	00001001	2 byte
486	00000100	Secondo dato: 1 byte
487	01001100	
488	01111100	Terzo dato:
489	01010101	4 byte
490	01001001	
491	01000111	Quarto dato:
492	01001001	2 byte
...	...	...

Codice ASCII

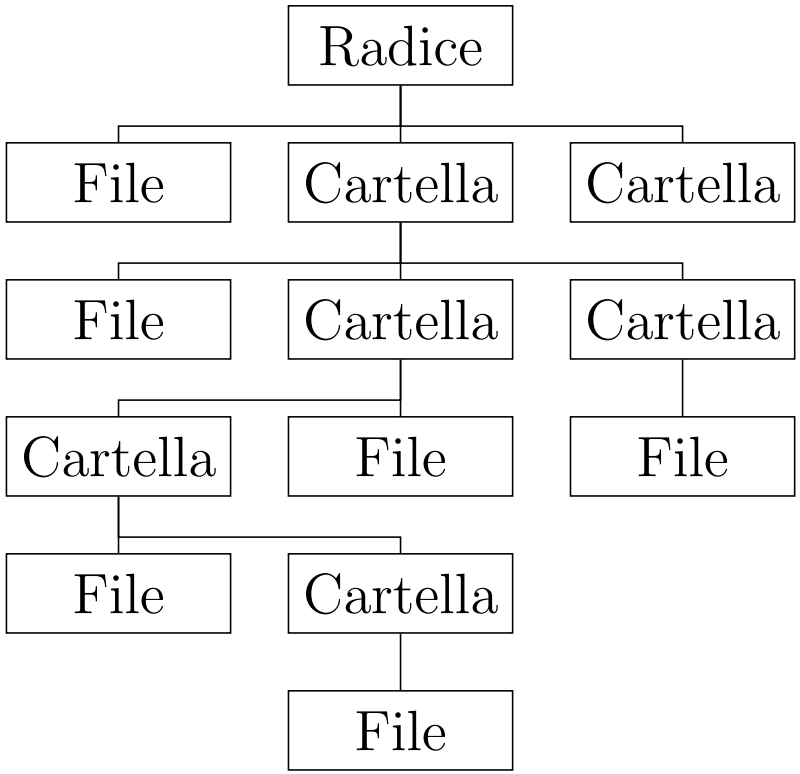
	000	001	010	011	100	101	110	111
0000		□	□	□	□	□	□	□
0001	□						□	□
0010	□	□	□	□	□	□	□	□
0011	□	□	□	□	□	□	□	□
0100		!	"	#	$	%	&	'
0101	(	)	*	+	,	-	.	/
0110	0	1	2	3	4	5	6	7
0111	8	9	:	;	<	=	>	?
1000	@	A	B	C	D	E	F	G
1001	H	I	J	K	L	M	N	O
1010	P	Q	R	S	T	U	V	W
1011	X	Y	Z	[	\	]	^	_
1100	‘	a	b	c	d	e	f	g
1101	h	i	j	k	l	m	n	o
1110	p	q	r	s	t	u	v	w
1111	x	y	z	{	|	}	~	□

Numerazione binaria

- Come in quella decimale, posizione di una cifra indica valore relativo
  - Il sistema binario usa potenze crescenti di 2

Binario	Decimale	Binario	Decimale
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

- Notazione realmente usata: complemento a due


Radice
File
Cartella
Cartella
File
Cartella
Cartella
Cartella
File
File
File
Cartella
File

Dalla teoria alla pratica

Esercitazione con shell Linux e prompt dei comandi Windows

Algoritmi e programmi

- **Informatica**: fusione di *informazione* e *automatica*
  - Studio sistematico degli algoritmi che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione
- **Algoritmo**: successione finita di istruzioni o passi che definiscono le operazioni da eseguire su dei dati (che formano l'istanza di un problema) per ottenere dei risultati (intesi come la soluzione dell'istanza specificata)
  - Interagisce con un ambiente esterno dal quale acquisisce dei dati e verso il quale comunica dati o messaggi

Equazione di primo grado

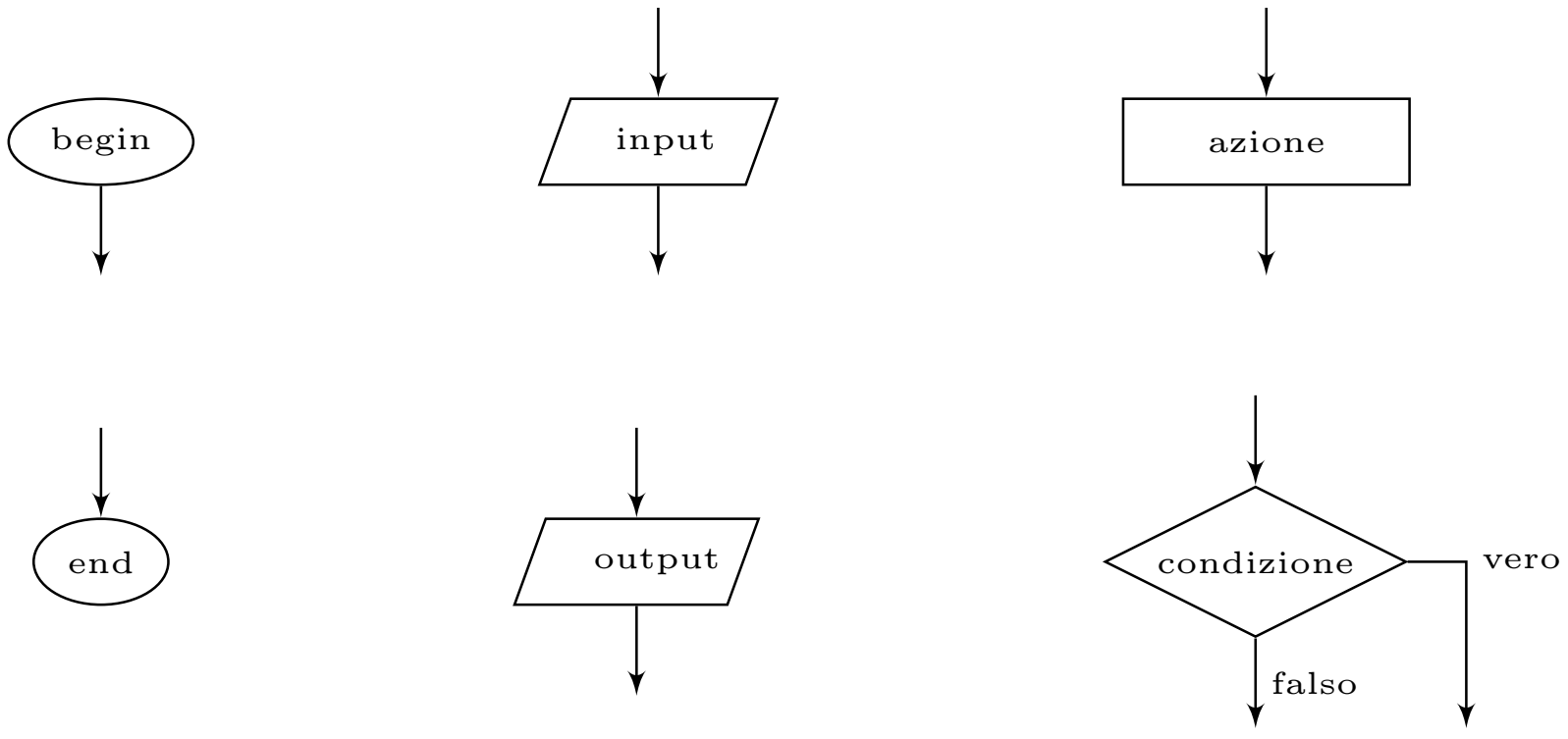
1. Inizio dell'algoritmo {
2. leggi i coefficienti a e b;
3. se $a \neq 0$, $x = -b/a$; vai a 6;
4. se $b \neq 0$, comunica che l'equazione è impossibile; vai a 7;
5. comunica che l'equazione è indeterminata; vai a 7;
6. comunica il valore di x;
7. } Fine dell'algoritmo

Somma di numeri interi

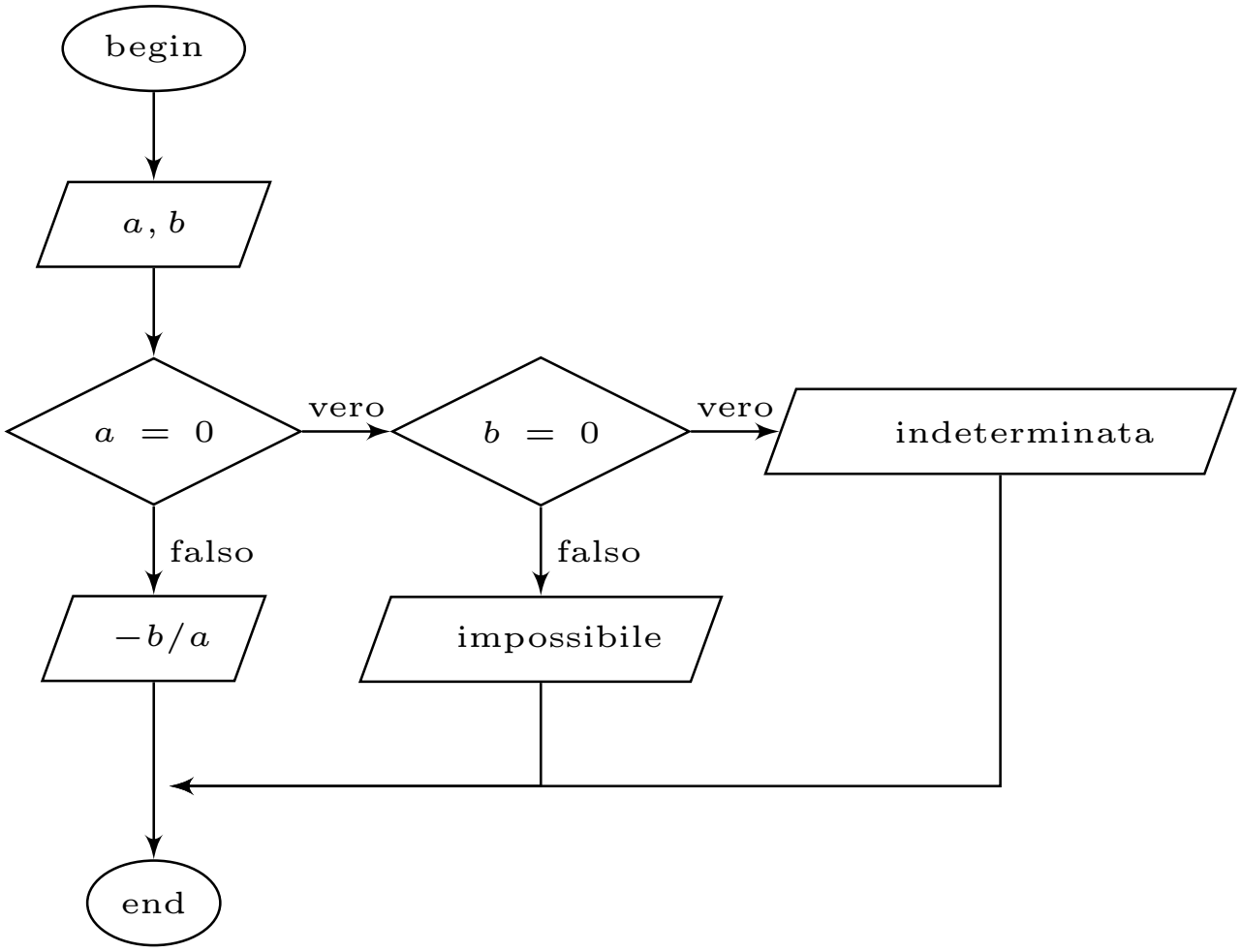
1. Inizio dell'algoritmo {
2. leggi n;
3. $i = 0$ e $s = 0$;
4. se $i > n$, vai a 6;
5. aggiungi i a s e incrementa i di 1; vai a 4;
6. comunica il valore di s;
7. } Fine dell'algoritmo

Descrizione degli algoritmi

- Linguaggio di programmazione
- Pseudo-codice
- Diagrammi a blocchi



Diagrammi a blocchi ed equazioni di primo grado

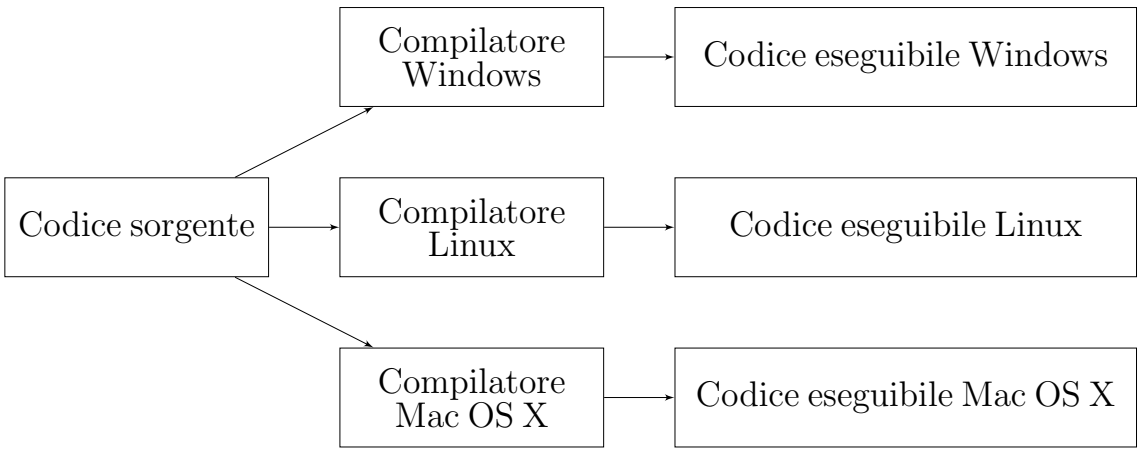


Programmi

- Programma: algoritmo scritto in linguaggio di programmazione
- Opera su
  - Dati in input
  - Dati di supporto
- Produce dati in output
- Diversi tipi
  - Sistema operativo
  - Programmi applicativi
    - Già esistenti
    - Creati dall'utente

- **Linguaggio macchina**
  - Consiste di sequenze di 0 e 1
  - Eseguito direttamente dal calcolatore
- **Linguaggio assembler**
  - Di tipo simbolico
  - Richiede una traduzione aggiuntiva molto semplice
- **Linguaggio ad alto livello**
  - Compromesso tra linguaggio naturale e linguaggio macchina
  - Esempi: FORTRAN, ALGOL, COBOL, LISP, APL, PROLOG, BASIC, Pascal, C, Ada, C++, Java, Python
- Esempio: somma di due numeri
  - Linguaggio macchina: `00000000001000001100000100000100000`
  - Linguaggio assembler: `add $3, $2, $1`
  - Linguaggio ad alto livello: `c = a+b;`

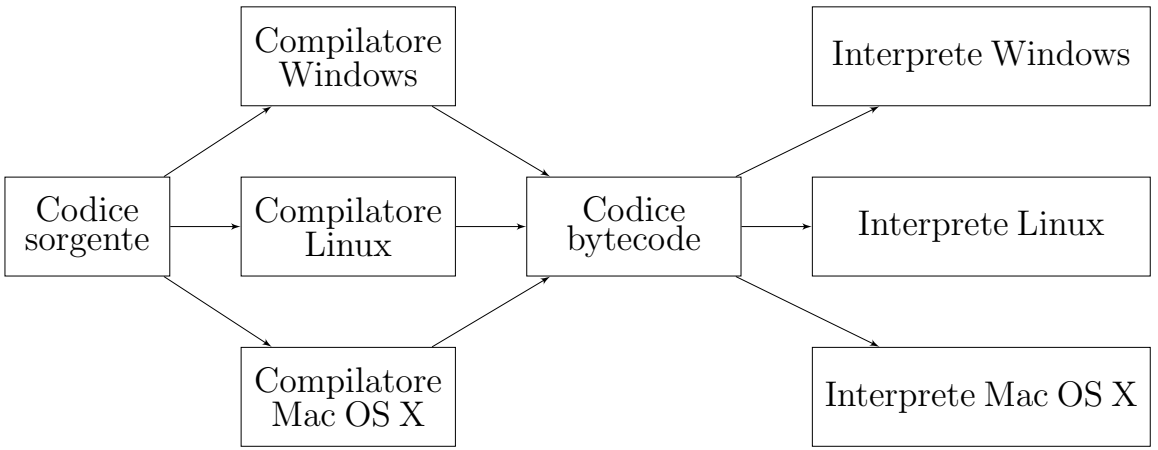
- **Compilatore**: traduce programma in linguaggio ad alto livello in programma in linguaggio macchina (più o meno)



- **Interprete**: traduce ed esegue una dopo l'altra istruzioni programma sorgente

L'approccio di Java

- Uso di codice intermedio detto **byte-code**
  - Linguaggio macchina di calcolatore virtuale



- Principale vantaggio: portabilità

Dalla teoria alla pratica

1. Compilazione programmi Java a linea di comando
2. Ambiente di sviluppo Eclipse