CAPITOLO

13

Algoritmi e dati

Un computer non capisce nulla di quello che fa: per questo le istruzioni che gli diamo devono essere perfette.

OBIETTIVI

1. Riconoscere un algoritmo e le sue proprietà: finitezza, non ambiguità, dati di ingresso e di uscita, generalità.
2. Descrivere un algoritmo con un diagramma di flusso e con lo pseudocodice, usando sequenza, selezione e iterazione.
3. Eseguire e controllare un algoritmo con la tabella di traccia; progettare semplici algoritmi numerici.
4. Collegare algoritmi e funzioni e comprendere il concetto di funzione calcolabile.
5. Spiegare come un computer rappresenta numeri interi, testi, immagini e suoni, e calcolare lo spazio che occupano.
6. Usare il foglio di calcolo per tabulare funzioni ed eseguire algoritmi.

INDICE DEL CAPITOLO

13.1 Che cos'è un algoritmo 2
13.2 Descrivere un algoritmo 3
13.3 Algoritmi e funzioni 9
13.4 Rappresentare i dati 10
13.5 Il foglio di calcolo come laboratorio 13

Matematica 1 · Parte VII · Elementi di informatica · versione 1.0
Capitolo 13: Algoritmi e dati · pagine 337–356 del volume completo · estratto per lo studio

Ogni volta che usi una calcolatrice, cerchi una strada sul navigatore o invii una foto, un computer esegue migliaia di istruzioni semplicissime, una dopo l'altra, senza capire che cosa sta facendo. Eppure il risultato è corretto. Il segreto è che qualcuno ha scritto quelle istruzioni in modo così preciso da non lasciare nulla all'interpretazione: ha scritto un *algoritmo*.

Hai già incontrato due algoritmi nel capitolo 1: quello di Euclide per il MCD e quello delle divisioni successive per scrivere un numero in base due. In questo capitolo l'algoritmo diventa l'oggetto di studio: vedremo come si descrive, come si controlla che funzioni, che rapporto ha con le funzioni del capitolo 5 e come un computer rappresenta i dati su cui lavora (numeri, testi, immagini, suoni). Chiuderemo con il foglio di calcolo, uno strumento che permette di eseguire algoritmi e di esplorare la matematica senza scrivere un vero programma.

13.1 Che cos'è un algoritmo

13.1.1 Istruzioni per un esecutore

Un **algoritmo** è una sequenza finita di istruzioni precise che, eseguite nell'ordine stabilito, risolvono un problema in un numero finito di passi. Le istruzioni sono rivolte a un **esecutore**: una persona, una macchina, un computer. L'esecutore non deve inventare nulla e non deve capire lo scopo del lavoro: deve soltanto saper eseguire ciascuna istruzione, come «dividi», «confronta», «scrivi».

Confronta due testi. Il primo è una ricetta: «aggiungi farina quanto basta e cuoci finché la torta è dorata». Il secondo è la regola del capitolo 1: «dividi il numero per 2, scrivi il resto, ripeti con il quoziente finché il quoziente è 0; leggi i resti dal basso verso l'alto». La ricetta funziona per un cuoco esperto, ma «quanto basta» e «dorata» non dicono a una macchina che cosa fare. La seconda regola, invece, può essere eseguita da chiunque, persino da chi non sa che cosa sia la base due, e dà sempre lo stesso risultato.

13.1.2 Le proprietà di un algoritmo

Perché un procedimento sia un algoritmo deve avere alcune caratteristiche.

- **Finitezza.** Le istruzioni sono in numero finito e l'esecuzione termina dopo un numero finito di passi, qualunque siano i dati.
- **Non ambiguità.** Ogni istruzione ha un solo significato possibile, e dopo ogni passo è stabilito qual è il passo successivo. Per questo, con gli stessi dati, l'algoritmo dà sempre lo stesso risultato: si dice che è **deterministico**.
- **Dati e risultati.** L'algoritmo riceve dei **dati di ingresso** (in inglese *input*) e produce dei **dati di uscita** (*output*), cioè il risultato.

- **Generalità.** Risolve non un singolo caso ma un'intera classe di problemi: l'algoritmo di Euclide calcola il MCD di *qualsiasi* coppia di numeri naturali non nulli, non soltanto di 252 e 198.

Rileggiamo con questi occhi l'algoritmo di Euclide. I dati di ingresso sono due naturali $a \geq b > 0$; il dato di uscita è il loro MCD. Ogni istruzione (calcola il resto, confronta il resto con 0, sostituisci) è precisa. E l'algoritmo termina sempre: i resti successivi sono numeri naturali sempre più piccoli, perché ogni resto è minore del divisore, e una successione di naturali che diminuiscono non può continuare all'infinito. Prima o poi un resto vale 0 e l'algoritmo si ferma.

ERRORE FREQUENTE

Un procedimento che non finisce non è un algoritmo. «Scrivi tutti i numeri pari» è un'istruzione chiarissima, ma non termina mai. Anche «parti da 1 e aggiungi 2 finché ottieni 100» non termina: 1, 3, 5, … sono tutti dispari e il valore 100 non viene mai raggiunto. Quando scrivi un ciclo, chiediti sempre: *sono sicuro che la condizione di arresto, prima o poi, si verificherà?*

APPROFONDIMENTO

Da al-Khwārizmī a «algoritmo»

La parola *algoritmo* deriva dal nome del matematico persiano Muḥammad ibn Mūsā al-Khwārizmī, attivo a Baghdad nel IX secolo. Un suo trattato sul calcolo con le cifre indiane fu tradotto in latino e il nome dell'autore, latinizzato in *Algoritmi*, finì per indicare il procedimento di calcolo stesso. Da un'altra sua opera, dedicata alla risoluzione delle equazioni, viene invece la parola *algebra*.

Gli algoritmi, però, sono molto più antichi della parola. Quello per il massimo comune divisore si legge già negli *Elementi* di Euclide (libro VII, proposizioni 1 e 2, intorno al 300 a.C.), dove procede togliendo ripetutamente il numero minore dal maggiore: Euclide lo descrive con il verbo ἀνθυφαιρεῖν, «togliere a vicenda», e per questo oggi lo si chiama ἀνθυφαίρεσις (*anthyphàiresis*), «sottrazione reciproca».

13.2 Descrivere un algoritmo

13.2.1 Variabili e assegnazione

Un algoritmo lavora su dei valori che cambiano durante l'esecuzione. Per riferirsi a essi si usano le **variabili**: nomi, come x, s o i, che indicano una «casella» della memoria in cui è conservato un valore. L'istruzione che mette un valore in una variabile si chiama **assegnazione** e si scrive con una freccia verso sinistra:

$$s \leftarrow 0 \quad i \leftarrow i + 1 \quad r \leftarrow \text{resto della divisione } a : b$$

La prima istruzione si legge «assegna a s il valore 0». La seconda si legge «calcola $i + 1$ e metti il risultato in i»: se prima i valeva 4, dopo vale 5. Il valore precedente di una variabile, quando le si assegna un nuovo valore, va perso.

ERRORE FREQUENTE

L'assegnazione non è un'equazione. Come equazione, $x = x + 1$ non ha soluzioni; come assegnazione, $x \leftarrow x + 1$ è un'istruzione sensatissima: aumenta di 1 il valore di x. Anche l'ordine conta: dopo $x \leftarrow 3$, $y \leftarrow x + 2$, $x \leftarrow 10$, la variabile y vale 5, non 12, perché la terza istruzione cambia x quando y è già stato calcolato.

13.2.2 Il diagramma di flusso

Un modo di descrivere un algoritmo è il **diagramma di flusso**, che hai già visto per l'algoritmo di Euclide nel capitolo 1. Ogni forma ha un significato preciso: l'ovale indica l'inizio e la fine, il parallelogramma la lettura dei dati o la scrittura dei risultati, il rettangolo un'istruzione, il rombo una domanda a cui si risponde sì o no. Le frecce indicano l'ordine in cui si procede (Figura 13.1).

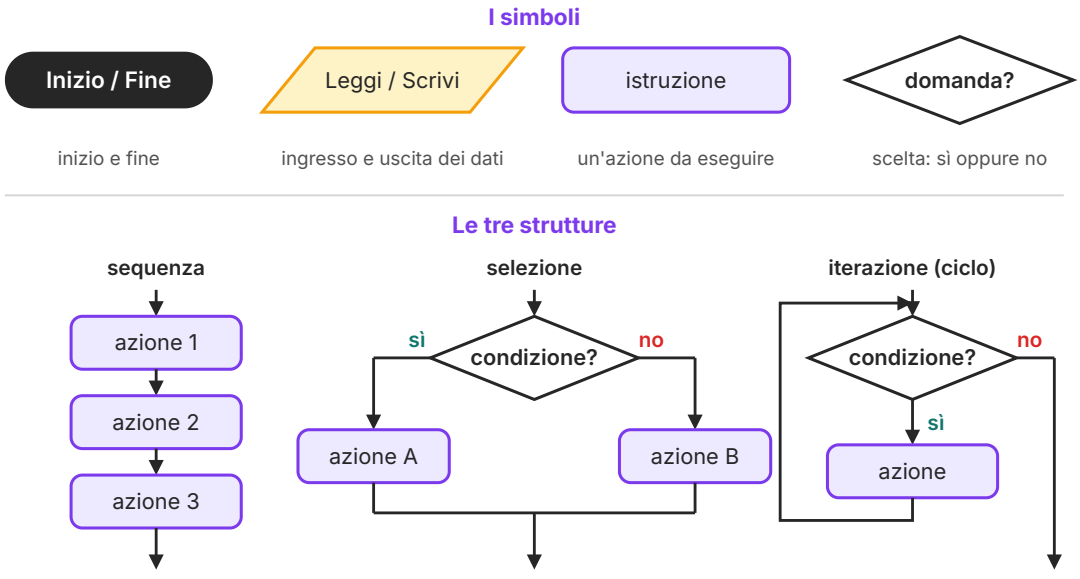


Figura 13.1 I simboli del diagramma di flusso e le tre strutture con cui si costruisce ogni algoritmo: la sequenza, la selezione e l'iterazione. Nell'iterazione la freccia che torna indietro fa ripetere l'azione finché la condizione è vera.

Con questi simboli si costruiscono tre schemi fondamentali. Nella **sequenza** le istruzioni si eseguono una dopo l'altra. Nella **selezione** una domanda fa scegliere tra due strade, che poi si ri-

congiungono. Nell'**iterazione**, o ciclo, un gruppo di istruzioni viene ripetuto finché una condizione resta vera; quando diventa falsa, si esce dal ciclo.

DA RICORDARE

Bastano tre strutture

Sequenza, selezione e iterazione bastano per descrivere qualsiasi algoritmo: è il contenuto di un teorema dimostrato nel 1966 da due informatici italiani, Corrado Böhm e Giuseppe Jacopini. Per questo i linguaggi di programmazione, pur diversissimi tra loro, hanno tutti un modo per eseguire istruzioni in sequenza, per scegliere (*if*) e per ripetere (*while*, *for*).

13.2.3 Lo pseudocodice

I diagrammi di flusso sono chiari ma ingombranti. Un'alternativa è lo **pseudocodice**: una descrizione a righe, in italiano, con poche parole chiave scritte in maiuscolo. In questo libro useremo LEGGI e SCRIVI per l'ingresso e l'uscita; SE … ALLORA … ALTRIMENTI … FINE SE per la selezione; MENTRE … ESEGUI … FINE MENTRE per l'iterazione. Le righe rientrate sono quelle che appartengono a una selezione o a un ciclo. Ecco un algoritmo che trova il massimo di tre numeri.

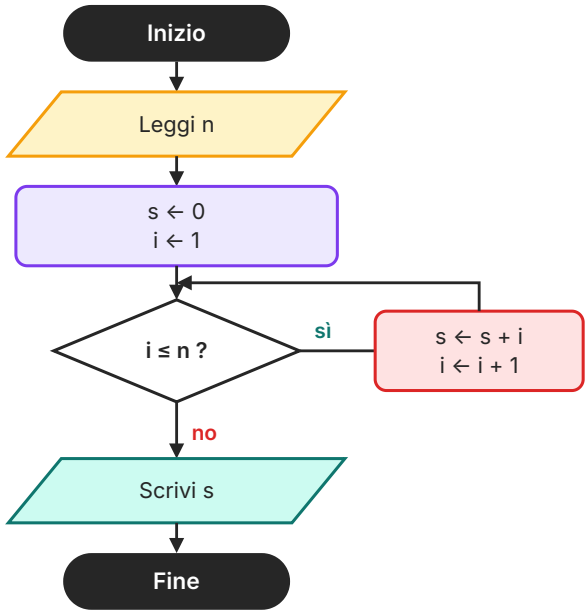
Pseudocodice Il massimo di tre numeri.

```
LEGGI a, b, c
m ← a
SE b > m ALLORA m ← b FINE SE
SE c > m ALLORA m ← c FINE SE
SCRIVI m
```

L'idea è quella che useresti a mano: tieni in m il «massimo visto finora» e lo aggiorni ogni volta che trovi un numero più grande. Con $a = 7$, $b = 12$, $c = 9$: m parte da 7, diventa 12 al primo confronto e resta 12 al secondo; l'algoritmo scrive 12.

13.2.4 La tabella di traccia

Come si controlla che un algoritmo faccia davvero quello che promette? Il modo più semplice è eseguirlo a mano su un esempio, annotando in una **tabella di traccia** il valore di ogni variabile dopo ogni passo. La figura 13.2 mostra un algoritmo con un ciclo, che calcola la somma $1 + 2 + \ldots + n$, e la sua traccia per $n = 4$.



Traccia con n = 4

passo	*i*	*s*
inizio	1	0
1	2	1
2	3	3
3	4	6
4	5	10

5 ≤ 4 è falso: il ciclo si ferma

uscita: s = 10 = 1 + 2 + 3 + 4

Figura 13.2 Diagramma di flusso dell'algoritmo che calcola la somma dei numeri naturali da 1 a n, e tabella di traccia per $n = 4$. Il ciclo (in rosso) si ripete finché $i \le n$.

ESEMPIO SVOLTO

Traccia e verifica con $n = 100$

Problema. Che cosa scrive l'algoritmo della figura 13.2 se $n = 100$? Quante volte viene eseguito il ciclo? Controlla il risultato con un altro metodo.

Soluzione. All'inizio $s = 0$ e $i = 1$. A ogni giro del ciclo si aggiunge i a s e si aumenta i di 1: dopo il primo giro $s = 1$, dopo il secondo $s = 1 + 2 = 3$, e così via. Il ciclo viene eseguito per $i = 1, 2, \ldots, 100$, cioè 100 volte; quando i diventa 101 la condizione $101 \le 100$ è falsa e l'algoritmo scrive $s = 1 + 2 + \ldots + 100$.

Per il controllo scriviamo la somma due volte, una in ordine crescente e una in ordine decrescente, e sommiamo in colonna: $1 + 100$, $2 + 99$, ..., $100 + 1$ sono 100 coppie, ciascuna di somma 101. Quindi due volte la somma vale $100 \cdot 101 = 10\,100$, e la somma vale **5050**. In generale $1 + 2 + \ldots + n = \dfrac{n(n+1)}{2}$.

13.2.5 Due algoritmi numerici

Un numero è primo? Per decidere se un naturale $n > 1$ è primo si può provare a dividerlo per 2, 3, 4, ...: se una divisione ha resto 0, n è composto. Non serve arrivare fino a $n - 1$. Se infatti $n = d \cdot e$ con $d \le e$, allora $d \cdot d \le d \cdot e = n$: un numero composto ha sempre un divisore d con

$d \cdot d \le n$. Basta quindi provare i divisori finché $d \cdot d \le n$. Questo metodo si chiama **prova dei divisori**.

Pseudocodice Prova dei divisori.

```
LEGGI n (naturale maggiore di 1)
d ← 2; primo ← vero
MENTRE d · d ≤ n e primo = vero ESEGUI
    SE il resto di n : d è 0 ALLORA primo ← falso FINE SE
    d ← d + 1
FINE MENTRE
SCRIVI primo
```

ESEMPIO SVOLTO

97 è primo?

Si prova con $d = 2, 3, \ldots, 9$, perché $9 \cdot 9 = 81 \le 97$ mentre $10 \cdot 10 = 100 > 97$. I resti di 97 diviso 2, 3, 4, 5, 6, 7, 8, 9 sono 1, 1, 1, 2, 1, 6, 1, 7: nessuno è 0. Dopo 8 divisioni l'algoritmo scrive «vero»: 97 è primo. Con il metodo ingenuo, che prova tutti i divisori da 2 a 96, le divisioni sarebbero state 95.

Approssimare √2. Nel capitolo 2 hai visto che nessun numero razionale ha quadrato uguale a 2. Possiamo però trovare numeri razionali il cui quadrato è vicino a 2 quanto vogliamo, con il **metodo di bisezione**. Si parte da un intervallo che contiene √2: poiché $1^2 = 1 < 2$ e $2^2 = 4 > 2$, √2 sta tra 1 e 2. Si calcola il punto medio m dell'intervallo: se $m^2 < 2$, √2 sta nella metà destra e m diventa il nuovo estremo sinistro; altrimenti sta nella metà sinistra e m diventa il nuovo estremo destro. A ogni passo l'ampiezza dell'intervallo si dimezza (figura 13.3, tabella 13.1).

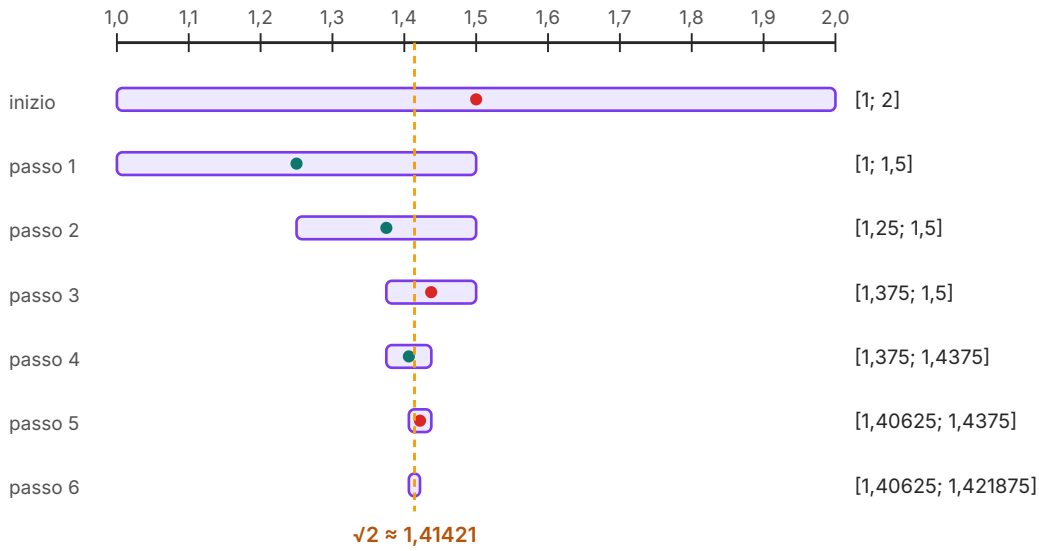


Figura 13.3 Il metodo di bisezione per √2. A ogni passo si calcola il punto medio dell'intervallo (in verde se il suo quadrato è minore di 2, in rosso se è maggiore) e si tiene la metà che contiene √2; l'ampiezza si dimezza.

Tabella 13.1 Le prime otto iterazioni della bisezione per √2 (i quadrati sono arrotondati a quattro decimali).

passo	a	b	$m = (a + b) : 2$	m^2	nuovo intervallo
1	1	2	1,5	> 2: 2,2500	[1; 1,5]
2	1	1,5	1,25	< 2: 1,5625	[1,25; 1,5]
3	1,25	1,5	1,375	< 2: 1,8906	[1,375; 1,5]
4	1,375	1,5	1,4375	> 2: 2,0664	[1,375; 1,4375]
5	1,375	1,4375	1,40625	< 2: 1,9775	[1,40625; 1,4375]
6	1,40625	1,4375	1,421875	> 2: 2,0217	[1,40625; 1,421875]
7	1,40625	1,421875	1,4140625	< 2: 1,9996	[1,4140625; 1,421875]
8	1,4140625	1,421875	1,41796875	> 2: 2,0106	[1,4140625; 1,41796875]

Dopo 8 passi sappiamo che 1,4140625 < √2 < 1,41796875: le prime due cifre decimali, 1,41, sono certe. Ogni passo aggiunge un bit di precisione; dopo 10 passi l'ampiezza è $1 : 2^{10} = 1 : 1024$, meno di un millesimo. L'algoritmo non trova mai √2 «esatto» (non potrebbe, perché √2 non è razionale), ma fornisce un'approssimazione con la precisione che si desidera: si fissa la precisione e ci si ferma quando l'intervallo è abbastanza piccolo. In questo modo anche la condizione di finitezza è rispettata.

13.3 Algoritmi e funzioni

13.3.1 Un algoritmo calcola una funzione

Nel capitolo 5 hai visto che una funzione associa a ogni elemento del dominio uno e un solo elemento del codominio. Un algoritmo deterministico fa qualcosa di molto simile: a ogni dato di ingresso fa corrispondere uno e un solo dato di uscita. L'algoritmo di Euclide, per esempio, realizza la funzione che alla coppia (a, b) associa $\text{MCD}(a, b)$; l'algoritmo della figura 13.2 realizza la funzione $n \mapsto 1 + 2 + \ldots + n$.

La funzione è il *che cosa* (quale uscita corrisponde a quale ingresso), l'algoritmo è il *come* (con quali operazioni la si ottiene). Una funzione si dice **calcolabile** se esiste un algoritmo che, per ogni dato del dominio, termina e restituisce il valore della funzione. Sono calcolabili tutte le funzioni che hai incontrato finora:

- la **parità**: a n associa «pari» se il resto di $n : 2$ è 0, «dispari» altrimenti;
- il **fattoriale** $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ (con $0! = 1$), per esempio $5! = 120$: si moltiplica in un ciclo, come nella figura 13.2 si somma;
- la **somma delle cifre** di n: si prende il resto della divisione per 10 (l'ultima cifra), lo si aggiunge alla somma, si sostituisce n con il quoziente e si ripete finché n diventa 0; per 2026 si ottiene $6 + 2 + 0 + 2 = 10$;
- le funzioni $x \mapsto ax + b$, $x \mapsto |x|$ e le altre del capitolo 5, almeno quando x è un numero razionale scritto come frazione.

APPROFONDIMENTO

Turing e il problema della fermata

Esistono funzioni che nessun algoritmo può calcolare? Per rispondere bisognava prima definire con precisione che cosa sia un algoritmo. Nel 1936 il matematico inglese Alan Turing propose un modello di esecutore molto semplice, oggi chiamato *macchina di Turing*: un nastro diviso in caselle, una testina che legge e scrive un simbolo alla volta, un elenco finito di regole. Tutto ciò che un computer di oggi sa calcolare, lo può calcolare anche una macchina di Turing (più lentamente).

Con questo modello Turing dimostrò che esistono problemi ben posti che nessun algoritmo risolve. Il più famoso è il **problema della fermata**: non esiste un algoritmo che, ricevuti un qualsiasi programma e i suoi dati, decida sempre correttamente se quel programma, eseguito su quei dati, prima o poi si ferma. La dimostrazione è per assurdo, come quella dell'irrazionalità di $\sqrt{2}$, e la studierai, se vorrai, all'università. La funzione che a ogni coppia (programma, dati) associa «si ferma» o «non si ferma» è quindi ben definita ma *non calcolabile*.

13.3.2 Stesso risultato, costo diverso

Due algoritmi diversi possono calcolare la stessa funzione con una quantità di lavoro molto diversa. Il numero di operazioni che un algoritmo esegue, al variare dei dati, ne misura il **costo**; un algoritmo che richiede meno operazioni è più efficiente.

- **La somma da 1 a *n*.** L'algoritmo con il ciclo esegue n addizioni (e n incrementi di i); la formula $n(n+1) : 2$ richiede tre operazioni, qualunque sia n. Per $n = 1\,000\,000$ la differenza è enorme.
- **Il MCD.** Per MCD(1071, 462) l'algoritmo di Euclide fa 3 divisioni: $1071 = 462 \cdot 2 + 147$, $462 = 147 \cdot 3 + 21$, $147 = 21 \cdot 7 + 0$, quindi MCD = 21. Il metodo della scomposizione richiede invece di scomporre entrambi i numeri, cosa che per numeri grandi è molto lenta.
- **La primalità.** Per un numero primo vicino a un milione, la prova dei divisori fino a $d \cdot d \le n$ fa circa mille divisioni; provare tutti i divisori fino a $n - 1$ ne richiederebbe quasi un milione.

Gli informatici studiano come cresce il costo di un algoritmo al crescere dei dati: è un capitolo importante dell'informatica, che qui ci limitiamo a nominare. Il messaggio da ricordare è che non basta che un algoritmo sia corretto: deve anche terminare in un tempo ragionevole per i dati con cui lo useremo.

13.4 Rappresentare i dati

13.4.1 I numeri interi

Nel capitolo 1 hai visto che i computer usano il sistema binario e che una cifra binaria si chiama bit. Un gruppo di 8 bit forma un **byte**, l'unità con cui si misura la memoria. Con n bit si scrivono 2^n sequenze diverse; se le usiamo per i naturali, rappresentiamo i numeri da 0 a $2^n - 1$. Con 8 bit si va da 0 a 255, con 16 bit da 0 a 65 535, con 32 bit da 0 a 4 294 967 295.

Che cosa succede se il risultato di un calcolo non sta nei bit disponibili? Su 8 bit, $255 = 11111111_2$; aggiungendo 1 si otterrebbe $256 = 100000000_2$, che ha 9 cifre. La nona cifra non trova posto e viene persa: il risultato memorizzato è 00000000, cioè 0. È come il contachilometri di un'auto vecchia che, dopo 99 999, torna a 00 000. Questo fenomeno si chiama **traboccamento** (*overflow*): su n bit un calcolo tra naturali dà il resto della divisione del risultato vero per 2^n. Per i numeri negativi e per quelli con la virgola si usano convenzioni apposite, che non vedremo.

13.4.2 I testi

Anche un testo, per un computer, è una sequenza di numeri: a ogni carattere si associa un numero, il suo codice. La prima tabella di codici diffusa in tutto il mondo fu l'**ASCII** (*American Standard Code for Information Interchange*), pubblicata nel 1963: usa 7 bit, e quindi $2^7 = 128$ codici,

per le lettere inglesi maiuscole e minuscole, le cifre, la punteggiatura, lo spazio e alcuni caratteri di controllo (tabella 13.2).

Tabella 13.2 Alcuni codici ASCII. Lettere e cifre hanno codici consecutivi.

Caratteri	Codici
spazio	32
0, 1, 2, ..., 9	48, 49, 50, ..., 57
A, B, C, ..., Z	65, 66, 67, ..., 90
a, b, c, ..., z	97, 98, 99, ..., 122
! ? .	33 63 46

Poiché le lettere hanno codici consecutivi, il codice di una lettera si ricava contando: la E, quinta lettera, ha codice 65 + 4 = 69. Ogni minuscola ha codice maggiore di 32 rispetto alla maiuscola corrispondente (a = 97 = 65 + 32), e il carattere «7» ha codice 48 + 7 = 55: attenzione, il *carattere* 7 non è il *numero* 7. La figura 13.4 mostra come la parola «Ciao!» diventa una sequenza di byte.

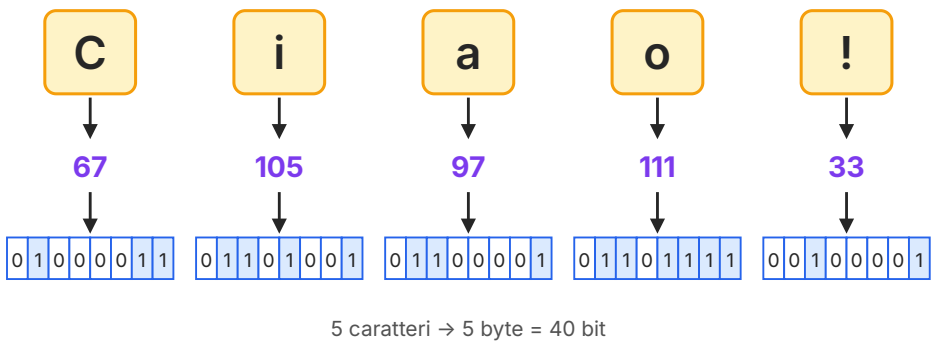


Figura 13.4 La parola «Ciao!» in codice ASCII: ogni carattere diventa un numero, e ogni numero diventa un byte. Le caselle azzurre sono le cifre 1.

L'ASCII non ha le lettere accentate, né l'alfabeto greco, né gli ideogrammi cinesi. Oggi si usa **Unicode**, una tabella unica che assegna un numero a più di 150 000 caratteri di quasi tutte le scritture del mondo, oltre a simboli matematici ed emoji; i primi 128 codici coincidono con l'ASCII. Il modo più diffuso di scrivere i codici Unicode in byte, UTF-8, usa da 1 a 4 byte per carattere: 1 byte per i caratteri ASCII, 2 per la «è» (codice 232), 3 per il simbolo dell'euro «€» (codice 8364), 4 per molte emoji. Un testo in italiano occupa quindi poco più di un byte per carattere.

ERRORE FREQUENTE

Lettere strane al posto degli accenti. Se un file scritto in UTF-8 viene letto con una vecchia tabella che usa un byte per carattere, ogni «è» diventa due caratteri, «Ã¨», perché i suoi due byte vengono letti come due caratteri separati. Non è un errore del testo, ma della tabella usata per leggerlo: i dati sono numeri, e il loro significato dipende dalla regola con cui li interpretiamo.

13.4.3 Immagini e suoni

Un'immagine digitale è una griglia di piccoli quadrati di colore uniforme, i **pixel** (da *picture element*). Il numero di pixel in orizzontale e in verticale è la risoluzione dell'immagine. Ogni colore si ottiene mescolando luce rossa, verde e blu (**codifica RGB**): per ciascuna delle tre componenti si usa un byte, con valori da 0 a 255, quindi 3 byte = 24 bit per pixel (figura 13.5). Le combinazioni possibili sono $256^3 = 2^{24} = 16\,777\,216$: più di quante il nostro occhio ne sappia distinguere.

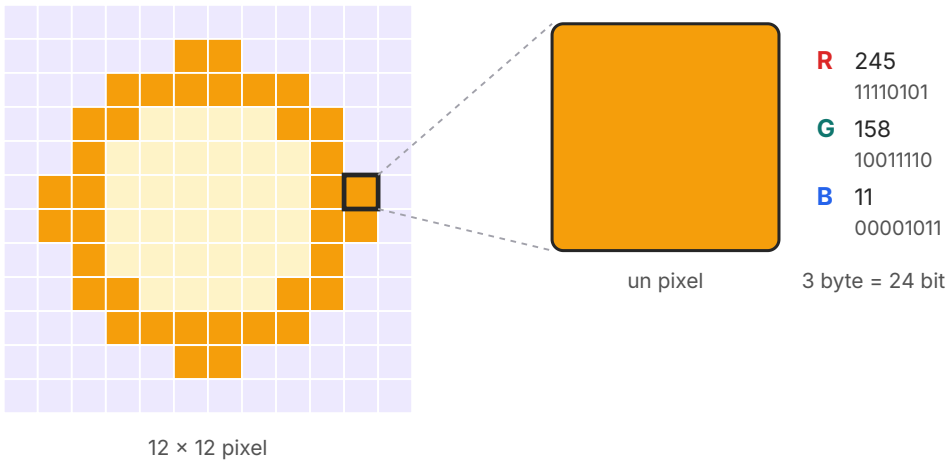


Figura 13.5 Un'immagine di 12 × 12 pixel e l'ingrandimento di uno dei pixel: il suo colore arancio è descritto da tre byte, uno per il rosso (245), uno per il verde (158) e uno per il blu (11).

Un suono è una vibrazione dell'aria che varia continuamente nel tempo. Per registrarlo, un microfono ne misura l'intensità a intervalli regolari: è il **campionamento**. Ogni misura, o campione, è un numero intero scritto con un certo numero di bit. Il formato dei CD audio usa 44 100 campioni al secondo, di 16 bit ciascuno, per ognuno dei due canali (stereo). Più campioni al secondo e più bit per campione danno un suono più fedele, ma un file più grande.

ESEMPIO SVOLTO

Quanto occupa?

Problema. Calcola lo spazio occupato, senza compressione, da a) una foto di 1920 × 1080 pixel a 24 bit; b) un minuto di musica in qualità CD.

Soluzione. a) I pixel sono 1920 · 1080 = 2 073 600; ciascuno occupa 3 byte, quindi 2 073 600 · 3 = 6 220 800 byte, circa 6,2 MB (un megabyte, MB, è un milione di byte). b) In un secondo ci sono 44 100 · 2 = 88 200 campioni (due canali), ciascuno di 2 byte: 176 400 byte al secondo. In un minuto: 176 400 · 60 = 10 584 000 byte, circa 10,6 MB.

I file reali sono di solito molto più piccoli, perché vengono *compressi*: algoritmi come quelli dei formati JPEG (per le foto) e MP3 (per la musica) eliminano le informazioni che l'occhio o l'orecchio quasi non percepiscono.

13.5 Il foglio di calcolo come laboratorio

13.5.1 Celle, formule e riferimenti

Un **foglio di calcolo** è una tabella le cui caselle, le celle, sono individuate da una lettera (la colonna) e da un numero (la riga): B3 è la cella nella colonna B e nella riga 3. In una cella si può scrivere un numero, un testo o una formula, che comincia con il segno «=» e calcola un valore a partire da altre celle. Nel capitolo 12 hai usato il foglio di calcolo per gli indici statistici; qui lo usiamo per eseguire algoritmi.

Il gesto più importante è il **trascinamento** di una formula verso il basso. Se in B2 scriviamo `=2*A2+1` e la copiamo in B3, il foglio la trasforma in `=2*A3+1`: il riferimento A2 è **relativo**, cioè significa «la cella della colonna A che sta sulla mia stessa riga». Così, scritti in A2, A3, … i valori di x, una sola formula trascinata produce la tabella dei valori della funzione $y = 2x + 1$. Se invece un riferimento deve restare fisso, lo si rende **assoluto** con il simbolo $: `$E$1` indica sempre la cella E1, ovunque si copi la formula.

IN SINTESI

Formule da ricordare (foglio di calcolo in italiano)

- `=A2+1` nella cella sotto un numero: genera 1, 2, 3, … trascinando in basso;
- `=$E$1*A2+$E$2`: valori di $y = mx + q$, con m in E1 e q in E2; cambiando E1 o E2 cambia tutta la tabella;
- `=RESTO(A2;B2)`: resto della divisione di A2 per B2; `=INT(A2/B2)`: quoziente intero;
- `=SE(condizione;valore se vera;valore se falsa)`: la selezione; per esempio `=SE(RESTO(A2;2)=0;"pari";"dispari")`;
- `=SOMMA(B2:B101)`, `=MEDIA(B2:B101)`, `=ARROTONDA(A2;3)` (arrotonda a 3 decimali).

Gli argomenti si separano con il punto e virgola; nelle versioni inglesi i nomi delle funzioni cambiano (MOD, IF, SUM…).

13.5.2 Un algoritmo riga per riga

Nel foglio di calcolo un ciclo diventa una sequenza di righe: ogni riga contiene i valori delle variabili dopo un giro, calcolati a partire dalla riga precedente. In pratica il foglio costruisce da solo la tabella di traccia.

ESEMPIO SVOLTO

L'algoritmo di Euclide in un foglio di calcolo

Problema. Organizza un foglio che calcoli MCD(1365, 1001) con l'algoritmo di Euclide.

Soluzione. Nella riga 2 scriviamo i dati: 1365 in A2, 1001 in B2, e in C2 la formula `=RESTO(A2;B2)`. Nella riga 3 la coppia successiva: `=B2` in A3 e `=C2` in B3 (il divisore diventa dividendo e il resto diventa divisore), poi copiamo C2 in C3. Trasciniamo le tre formule della riga 3 verso il basso e otteniamo la tabella 13.3. Il MCD è il valore in colonna B nella riga in cui il resto diventa 0: **MCD = 91**. Controllo: $1365 = 3 \cdot 5 \cdot 7 \cdot 13$ e $1001 = 7 \cdot 11 \cdot 13$, quindi $\text{MCD} = 7 \cdot 13 = 91$.

Tabella 13.3 Il foglio che esegue l'algoritmo di Euclide per 1365 e 1001. Nella riga 6 il divisore è 0 e la formula del resto darebbe un errore: lì l'algoritmo è già finito.

	A	B	C
1	dividendo	divisore	resto
2	1365	1001	364
3	1001	364	273
4	364	273	91
5	273	91	0

Nello stesso modo si esegue la bisezione per √2: in A2 e B2 gli estremi 1 e 2, in C2 il punto medio `=(A2+B2)/2`, in D2 il suo quadrato `=C2^2`; nella riga 3, `=SE(D2<2;C2;A2)` in A3 e `=SE(D2<2;B2;C2)` in B3. Trascinando per qualche decina di righe si ritrovano i valori della tabella 13.1 e, molto più in basso, tutte le cifre di √2 che il foglio è in grado di mostrare.

COLLEGAMENTI

Non solo numeri. Il foglio di calcolo è uno dei programmi che permettono di rappresentare e manipolare oggetti matematici. Altri sono i programmi di geometria dinamica, con cui nei capitoli di geometria puoi eseguire le costruzioni con riga e compasso e poi muovere i punti per vedere che cosa resta vero, e i programmi che disegnano i grafici delle funzioni del capitolo 5. Usali come laboratorio: per fare congetture, per controllare un calcolo, per provare molti casi. Ma ricorda che il computer esegue, non dimostra: un risultato che vale in cento casi provati non è ancora un teorema.

Con questo capitolo hai visto che dietro ogni programma c'è un'idea matematica precisa: un algoritmo che trasforma dati in risultati, e una codifica che trasforma testi, immagini e suoni in numeri. Nel corso degli anni di liceo incontrerai altri algoritmi, per risolvere equazioni, approssimare aree, simulare esperimenti casuali, e li potrai sempre descrivere, eseguire e controllare con gli strumenti di queste pagine.

SINTESI

- Un algoritmo è una sequenza finita di istruzioni precise, rivolte a un esecutore, che risolve una classe di problemi in un numero finito di passi.
- Proprietà: finitezza, non ambiguità (determinismo), dati di ingresso e di uscita, generalità.
- Le variabili contengono valori; l'assegnazione $x \leftarrow x + 1$ sostituisce il valore di x e non è un'equazione.
- Diagramma di flusso: ovale (inizio/fine), parallelogramma (ingresso/uscita), rettangolo (istruzione), rombo (domanda sì/no). Pseudocodice: LEGGI, SCRIVI, SE … ALLORA … ALTRIMENTI, MENTRE … ESEGUI.
- Ogni algoritmo si costruisce con sequenza, selezione e iterazione (teorema di Böhm e Jacopini).
- La tabella di traccia registra il valore delle variabili passo per passo e serve a controllare un algoritmo.
- Per sapere se n è primo basta provare i divisori d con $d \cdot d \le n$. La bisezione approssima $\sqrt{2}$ dimezzando a ogni passo un intervallo che lo contiene.
- Un algoritmo deterministico realizza una funzione; una funzione è calcolabile se esiste un algoritmo che la calcola. Esistono funzioni non calcolabili (Turing, problema della fermata).
- Algoritmi diversi per la stessa funzione possono avere costi molto diversi.
- Con n bit si rappresentano i naturali da 0 a $2^n - 1$; oltre si ha traboccamento. ASCII: 7 bit, A = 65, a = 97, «0» = 48. Unicode/UTF-8: da 1 a 4 byte per carattere.
- Immagini: pixel e RGB (3 byte per pixel). Suoni: campionamento (CD: 44 100 campioni/s, 16 bit, 2 canali).
- Foglio di calcolo: riferimenti relativi (A2) e assoluti ($A$1); trascinando una formula si esegue un ciclo riga per riga.

ESERCIZI

Comprendere

1. ● Quale dei seguenti testi **non** descrive un algoritmo?
   A. Dividi il numero per 2 e scrivi il resto; ripeti con il quoziente finché il quoziente è 0.
   B. Aggiungi farina quanto basta e mescola finché l'impasto è morbido.
   C. Confronta i primi due numeri e tieni il maggiore; confrontalo con il terzo e tieni il maggiore.
   D. Per ogni numero da 1 a 10 scrivi il suo quadrato.

2. ● In un diagramma di flusso, il rombo indica:
   A. un'istruzione da eseguire
   B. la lettura dei dati
   C. una domanda a cui si risponde sì o no
   D. l'inizio o la fine

3. ● Dopo le istruzioni $x \leftarrow 5$; $y \leftarrow x + 2$; $x \leftarrow y \cdot 2$ le variabili valgono:
   A. $x = 5$, $y = 7$
   B. $x = 10$, $y = 7$
   C. $x = 14$, $y = 14$
   D. $x = 14$, $y = 7$

4. ●● Con 10 bit si possono rappresentare i numeri naturali:
   A. da 0 a 1023
   B. da 0 a 10
   C. da 1 a 1000
   D. da 0 a 100

5. ●● Nel codice ASCII la lettera A ha codice 65. Il codice della lettera E è:
   A. 68
   B. 70
   C. 69
   D. 101

6. ●● Un'immagine di 100×200 pixel a 24 bit occupa, senza compressione:
   A. 20 000 byte
   B. 60 000 byte
   C. 480 000 byte
   D. 60 000 bit

7. ● Vero o falso? Correggi le affermazioni false.
a) Un algoritmo deve terminare dopo un numero finito di passi. b) $x \leftarrow x + 1$ è un'equazione impossibile. c) In UTF-8 ogni carattere occupa esattamente un byte. d) Con 8 bit si rappresentano 256 numeri diversi. e) Nel foglio di calcolo, copiando la formula `=A2*2` nella cella sottostante, si ottiene `=A3*2`.

8. ● Spiega con parole tue la differenza tra una funzione e un algoritmo che la calcola. Fai l'esempio della somma $1 + 2 + \ldots + n$.

Eseguire un algoritmo

9. ● Esegui l'algoritmo della figura 13.2 con $n = 6$, compilando la tabella di traccia. Quante volte si esegue il ciclo? Controlla il risultato con la formula di Gauss.

10. ● Esegui l'algoritmo seguente con $n = 5$ e con $n = 1$. Che cosa calcola?
LEGGI n; $p \leftarrow 1$; $i \leftarrow 1$; MENTRE $i \le n$ ESEGUI $p \leftarrow p \cdot i$; $i \leftarrow i + 1$ FINE MENTRE; SCRIVI p.

11. ●● Esegui con $n = 2026$, compilando la tabella di traccia, l'algoritmo seguente, e spiega che cosa calcola.
LEGGI n; $s \leftarrow 0$; MENTRE $n > 0$ ESEGUI $s \leftarrow s$ + (resto di n : 10); $n \leftarrow$ quoziente intero di n : 10 FINE MENTRE; SCRIVI s.

12. ●● Esegui quattro passi del metodo di bisezione per approssimare $\sqrt{3}$, partendo dall'intervallo $[1; 2]$. Scrivi a ogni passo il punto medio, il suo quadrato e il nuovo intervallo.

13. ●● Calcola MCD(2431, 1309) con l'algoritmo di Euclide, scrivendo tutte le divisioni. Quante divisioni servono? Controlla con la scomposizione in fattori primi.

14. ●● Usa la prova dei divisori ($d \cdot d \le n$) per stabilire se 221 e 223 sono primi. Quante divisioni esegue l'algoritmo in ciascun caso?

Costruire un algoritmo

15. ● Scrivi in pseudocodice e disegna il diagramma di flusso di un algoritmo che legge due numeri e scrive il maggiore (se sono uguali, scrive uno dei due).

16. ● Scrivi un algoritmo che legge un naturale n e scrive «pari» o «dispari», usando il resto della divisione per 2.

17. ●● Scrivi un algoritmo che legge n e scrive la somma dei primi n numeri dispari $1 + 3 + 5 + \ldots$. Eseguilo con $n = 1, 2, 3, 4$: che cosa noti?

18. ●● Scrivi in pseudocodice l'algoritmo delle divisioni successive per convertire un naturale in base 2, ed eseguilo con 26 e con 45 (confronta con il capitolo 1).

19. ●● Scrivi un algoritmo che legge due naturali non nulli e scrive il loro mcm, usando l'algoritmo di Euclide e la relazione MCD · mcm = $a \cdot b$. Eseguilo con 12 e 18.

20. ●●● Scrivi un algoritmo che legge n voti e scrive la media, il voto più alto e il voto più basso, usando un solo ciclo. Eseguilo con i voti 6, 8, 5, 7.

Rappresentare i dati

21. ● Scrivi in binario, con 8 bit ciascuno, i codici ASCII della parola «OK» (O ha codice 79, K ha codice 75).

22. ●● Decodifica la sequenza di byte ASCII 01001101 01100001 01110100 01100101.

23. ●● Quanti bit servono, come minimo, per dare un numero diverso a ciascuno dei 1000 posti di una sala? E per 5000 posti?

24. ●● Una foto di 4000 × 3000 pixel a 24 bit è salvata in un file JPEG di 3,6 MB. Quanto occuperebbe senza compressione? Di quante volte la compressione ha ridotto il file?

Con il computer

25. ●● Con un foglio di calcolo somma i numeri da 1 a 100 in due modi: con una colonna di numeri e la funzione SOMMA, e con la formula di Gauss in una sola cella. Scrivi le formule usate.

26. ●●● Con un foglio di calcolo trova il più piccolo esponente n per cui 2^n supera un milione. Che cosa significa il risultato per il numero di bit necessari a rappresentare i numeri fino a un milione?

Verso la verifica

27. ●● Spiega perché l'algoritmo di Euclide termina sempre, qualunque siano i due numeri di partenza.

28. ●●● Due algoritmi stabiliscono se n è primo: l'algoritmo A prova tutti i divisori da 2 a $n - 1$; l'algoritmo B prova i divisori d finché $d \cdot d \le n$. a) Dimostra che B è corretto, cioè che se n è composto ha un divisore d con $2 \le d$ e $d \cdot d \le n$. b) Il numero 10 007 è primo: quante divisioni eseguono A e B?

PER APPROFONDIRE

- [POL] Le strategie per risolvere un problema, che sono anche le prime idee per progettare un algoritmo.
- [CR] Contiene l'algoritmo di Euclide e una discussione dei numeri irrazionali e delle loro approssimazioni.
- [EUC] Il libro VII degli *Elementi* descrive il procedimento delle sottrazioni successive per trovare la «misura comune» di due numeri: è uno dei più antichi algoritmi ancora in uso.

Glossario del capitolo

Algoritmo
Sequenza finita di istruzioni precise che, eseguite in ordine, risolve una classe di problemi in un numero finito di passi.

ASCII
Codifica a 7 bit di 128 caratteri: lettere inglesi, cifre, punteggiatura e caratteri di controllo (A = 65, a = 97).

Assegnazione
Istruzione $x \leftarrow$ valore, che mette un valore in una variabile cancellando quello precedente.

Campionamento
Misura di un segnale, come un suono, a intervalli di tempo regolari, per rappresentarlo con numeri.

Deterministico
Detto di un algoritmo in cui ogni passo è stabilito senza ambiguità: con gli stessi dati dà sempre lo stesso risultato.

Diagramma di flusso
Rappresentazione grafica di un algoritmo con ovali (inizio/fine), parallelogrammi (ingresso/uscita), rettangoli (istruzioni), rombi (domande) e frecce.

Esecutore
Chi (persona o macchina) esegue le istruzioni di un algoritmo, senza bisogno di conoscerne lo scopo.

Funzione calcolabile
Funzione per cui esiste un algoritmo che, per ogni dato del dominio, termina e restituisce il valore della funzione.

Iterazione
Struttura che ripete un gruppo di istruzioni finché una condizione resta vera; detta anche ciclo.

Metodo di bisezione
Algoritmo che approssima un numero dimezzando a ogni passo un intervallo che lo contiene.

Pixel
Ciascuno dei quadratini di colore uniforme che formano un'immagine digitale; in codifica RGB occupa 3 byte.

Pseudocodice
Descrizione di un algoritmo a righe, in linguaggio naturale con parole chiave come LEGGI, SCRIVI, SE, MENTRE.

Riferimento assoluto
In un foglio di calcolo, riferimento a una cella che non cambia quando la formula viene copiata (per esempio $A$1).

Selezione
Struttura in cui una condizione fa scegliere quale gruppo di istruzioni eseguire.

Tabella di traccia
Tabella che registra il valore delle variabili di un algoritmo dopo ogni passo dell'esecuzione.

Traboccamento
(*overflow*) Situazione in cui il risultato di un calcolo non può essere rappresentato con i bit disponibili.

Unicode
Codifica che assegna un numero a ogni carattere delle scritture del mondo; in UTF-8 ogni carattere occupa da 1 a 4 byte.

Variabile (informatica)
Nome che indica una posizione della memoria in cui è conservato un valore, che può cambiare durante l'esecuzione.

Soluzioni

1. B. «Quanto basta» e «morbido» sono ambigui: un esecutore non sa quando fermarsi.

2. C.

3. D. Dopo la seconda istruzione $y = 7$; la terza dà $x = 7 \cdot 2 = 14$.

4. A. $2^{10} = 1024$ sequenze: da 0 a 1023.

5. C. 65 + 4 = 69 (E è la quinta lettera).

6. B. 100 · 200 = 20 000 pixel, 3 byte ciascuno: 60 000 byte.

7. a) Vero. b) Falso: è un'assegnazione, che aumenta di 1 il valore di x. c) Falso: da 1 a 4 byte (1 per i caratteri ASCII, 2 per «è»). d) Vero: da 0 a 255. e) Vero: il riferimento A2 è relativo.

8. La funzione dice quale uscita corrisponde a ogni ingresso; l'algoritmo dice come ottenerla. La funzione $n \mapsto 1 + 2 + \ldots + n$ si può calcolare con un ciclo che fa n addizioni, oppure con la formula $n(n + 1) : 2$: due algoritmi diversi, una sola funzione (per $n = 10$ entrambi danno 55).

9. Traccia (i, s): (1, 0), (2, 1), (3, 3), (4, 6), (5, 10), (6, 15), (7, 21). Il ciclo si esegue 6 volte; quando $i = 7$ la condizione $7 \le 6$ è falsa e l'algoritmo scrive 21. Formula: $6 \cdot 7 : 2 = 21$.

10. Con $n = 5$: p vale 1, 1, 2, 6, 24, 120; scrive 120. Con $n = 1$ scrive 1. Calcola il fattoriale $n! = 1 \cdot 2 \cdot \ldots \cdot n$.

11. Traccia (n, s): (2026, 0), (202, 6), (20, 8), (2, 8), (0, 10). Scrive 10: è la somma delle cifre di n, perché il resto della divisione per 10 è l'ultima cifra e il quoziente intero è il numero senza l'ultima cifra.

12. Passo 1: $m = 1{,}5$, $m^2 = 2{,}25 < 3 \to [1{,}5; 2]$. Passo 2: $m = 1{,}75$, $m^2 = 3{,}0625 > 3 \to [1{,}5; 1{,}75]$. Passo 3: $m = 1{,}625$, $m^2 = 2{,}640625 < 3 \to [1{,}625; 1{,}75]$. Passo 4: $m = 1{,}6875$, $m^2 = 2{,}84765625 < 3 \to [1{,}6875; 1{,}75]$. Quindi $1{,}6875 < \sqrt{3} < 1{,}75$ ($\sqrt{3} \approx 1{,}732$).

13. 2431 = 1309 · 1 + 1122; 1309 = 1122 · 1 + 187; 1122 = 187 · 6 + 0. Servono 3 divisioni: MCD = 187. Controllo: 2431 = 11 · 13 · 17, 1309 = 7 · 11 · 17, MCD = 11 · 17 = 187.

14. 221: si provano 2, 3, ..., 13; 221 : 13 = 17 con resto 0, quindi 221 = 13 · 17 non è primo (12 divisioni). 223: 14 · 14 = 196 ≤ 223 e 15 · 15 = 225 > 223, quindi si provano 2, 3, ..., 14 (13 divisioni), tutte con resto diverso da 0: 223 è primo.

15. LEGGI a, b; SE $a \ge b$ ALLORA SCRIVI a ALTRIMENTI SCRIVI b FINE SE. Diagramma: Inizio → Leggi a, b → rombo «$a \ge b$?» → sì: Scrivi a; no: Scrivi b → Fine.

16. LEGGI n; $r \leftarrow$ resto di $n : 2$; SE $r = 0$ ALLORA SCRIVI «pari» ALTRIMENTI SCRIVI «dispari» FINE SE.

17. LEGGI n; $s \leftarrow 0$; $i \leftarrow 1$; MENTRE $i \le n$ ESEGUI $s \leftarrow s + (2i - 1)$; $i \leftarrow i + 1$ FINE MENTRE; SCRIVI s. Risultati: 1, 4, 9, 16: la somma dei primi n dispari è n^2.

18. LEGGI n; MENTRE $n > 0$ ESEGUI SCRIVI resto di $n : 2$; $n \leftarrow$ quoziente intero di $n : 2$ FINE MENTRE; le cifre scritte si leggono dall'ultima alla prima. 26: resti 0, 1, 0, 1, 1 → 11010_2. 45: resti 1, 0, 1, 1, 0, 1 → 101101_2.

19. LEGGI a, b; $p \leftarrow a \cdot b$; MENTRE $b \ne 0$ ESEGUI $r \leftarrow$ resto di $a : b$; $a \leftarrow b$; $b \leftarrow r$ FINE MENTRE; SCRIVI $p : a$ (alla fine a contiene il MCD). Con 12 e 18: $p = 216$; coppie (12, 18), (18, 12), (12, 6), (6, 0): MCD = 6; mcm = 216 : 6 = 36. (Se $a < b$ il primo giro scambia i due numeri.)

20. LEGGI n; LEGGI v; $s \leftarrow v$; max $\leftarrow v$; min $\leftarrow v$; $i \leftarrow 2$; MENTRE $i \le n$ ESEGUI LEGGI v; $s \leftarrow s + v$; SE $v >$ max ALLORA max $\leftarrow v$ FINE SE; SE $v <$ min ALLORA min $\leftarrow v$ FINE SE; $i \leftarrow i + 1$ FINE MENTRE; SCRIVI $s : n$, max, min. Con 6, 8, 5, 7: somma 26, media 6,5, massimo 8, minimo 5.

21. 79 = 64 + 8 + 4 + 2 + 1 = 01001111; 75 = 64 + 8 + 2 + 1 = 01001011.

22. 77, 97, 116, 101, cioè M, a, t, e: «Mate».

23. $2^9 = 512 < 1000 \le 2^{10} = 1024$: servono 10 bit. Per 5000 posti: $2^{12} = 4096 < 5000 \le 2^{13} = 8192$, servono 13 bit.

24. 4000 · 3000 = 12 000 000 pixel; · 3 byte = 36 000 000 byte = 36 MB. 36 : 3,6 = 10: il file è 10 volte più piccolo.

25. A1: 1; A2: `=A1+1` trascinata fino ad A100; in una cella libera `=SOMMA(A1:A100)`, che dà 5050. In un'altra cella `=100*(100+1)/2`, che dà ancora 5050.

26. In A1 1, in A2 `=A1+1` trascinata; in B1 `=2^A1` trascinata. $2^{19} = 524\,288 < 1\,000\,000 < 2^{20} = 1\,048\,576$: $n = 20$. Per rappresentare in binario tutti i naturali fino a un milione servono 20 bit (con 19 si arriva solo a 524 287).

27. A ogni passo il nuovo divisore è il resto della divisione precedente, che è minore del divisore: i divisori formano una successione di numeri naturali strettamente decrescente. Una tale successione non può continuare all'infinito (al massimo arriva a 0 dopo un numero di passi non superiore al primo divisore), quindi prima o poi un resto vale 0 e l'algoritmo si ferma.

28. a) Se n è composto, $n = d \cdot e$ con $2 \le d \le e$ (chiamiamo d il minore dei due fattori). Moltiplicando $d \le e$ per d si ha $d \cdot d \le d \cdot e = n$. Quindi, se nessun d con $d \cdot d \le n$ divide n, n è primo. b) A esegue 10 005 divisioni (da 2 a 10 006). B: $100 \cdot 100 = 10\,000 \le 10\,007$ e $101 \cdot 101 = 10\,201 > 10\,007$, quindi prova da 2 a 100: 99 divisioni, circa cento volte di meno.