Prerequisiti:

- Conoscenza dei numeri naturali e interi e delle loro proprietà.
- Acquisizione del concetto di funzione.

Questa unità è riservata al primo biennio dei Licei, eccezion fatta per il Liceo Scientifico – opzione scienze applicate.

OBIETTIVI DI APPRENDIMENTO

Una volta completata l'unità, gli allievi devono essere in grado di:

- *fornire una definizione non formalizzata di algoritmo*
- *definire e applicare in casi semplici le strutture" sequenza", "scelta", "ciclo"*
- *elaborare un algoritmo risolutivo di qualche semplice problema*
- *fornire una definizione non formalizzata di funzione calcolabile*
- *fornire semplici esempi di funzioni calcolabili*

28.1 Algoritmo.

28.2 Notazione lineare strutturata.

28.3 Funzioni calcolabili.

Verifiche.

Una breve sintesi per domande e risposte.

Algoritmo. Funzioni calcolabili.

Unità 28

28.1 ALGORITMO

28.1.1 La gestione consapevole di una calcolatrice, di un foglio elettronico, di un software matematico, ma anche la capacità di gestire un programma di video-scrittura o il modo di navigare in Internet si basano su sequenze ordinate e finite di istruzioni che un "automa" - cioè una macchina automatica costruita per funzionare in base ad un programma – esegue nel modo in cui sono state ideate da un progettista. O, come anche si dice, sulla base di opportuni *algoritmi*.

Noi non ci occuperemo del modo in cui si progetta un automa, un software o anche del modo di scrivere un programma codificato in un qualunque linguaggio di programmazione: si tratta di un compito per specialisti.

Ci soffermeremo invece sugli algoritmi, anche perché questi non sono nati esattamente con l'informatica ma proprio con la matematica allo scopo di risolvere problemi, benché sia stata di recente l'informatica a valorizzarne e approfondirne l'uso.

La parola ***algoritmo*** deriva dal nome del matematico arabo **Mohammed ibn-Musa al-Khuwarizmi** (IX sec.) ma in un modo piuttosto singolare.

Questo matematico scrisse alcune opere di matematica e astronomia. Una di esse fece conoscere agli Europei il sistema decimale posizionale degli Indiani e ci pervenne in una traduzione latina del XII secolo col titolo *Algorithmi de numero Indorum*, la cui traduzione letterale è *Sul numero degli Indiani di al-Khuwarizmi*. Quest'opera introduceva dunque per la prima volta il termine "algoritmo", riferito però semplicemente e senza altri sottintesi all'autore dell'opera, appunto al-Khuwarizmi. In poco tempo, tuttavia, assunse il significato ben preciso di "metodo sistematico di calcolo".

28.1.2 Un algoritmo è dunque un procedimento di calcolo. Al fine di averne un'idea sufficientemente corretta, consideriamo il problema della ricerca del Massimo Comune Divisore (MCD) di due numeri.

Lo si può trovare, com'è noto, ricorrendo ad una regola basata sulla fattorizzazione dei numeri.

Per esempio, se i numeri sono 36 e 24, dapprima si trova che: $36=2^2\cdot 3^2$ e $24=2^3\cdot 3$. Il MCD è dato dal prodotto dei fattori comuni presi con il minore esponente. Perciò: $\mathrm{MCD}(36, 24)=2^2\cdot 3=12$.

Vogliamo proporre adesso un nuovo procedimento, noto come ***algoritmo euclideo delle divisioni successive***, perché fu proposto già da Euclide nei suoi *Elementi* (libro VII, prop. 2), anche se il grande matematico alessandrino non usava ovviamente il termine "algoritmo". Applicando tale metodo, si calcola anzitutto il resto della divisione del numero 36 per 24: si ottiene il resto 12. Siccome (cosa che è possibile dimostrare) ogni divisore comune ai due numeri 36 e 24 è divisore anche del resto 12, possiamo affermare che:

$$\mathrm{MCD}(36, 24) = \mathrm{MCD}(24, 12).$$

Ripetiamo l'operazione sostituendo ai numeri 36 e 24 rispettivamente 24 e 12. Troviamo dunque il resto della divisione di 24 per 12: questa volta il resto è 0. Ciò significa che 12 è divisore di 24 e, soprattutto, 12 è il massimo comune divisore dal momento che evidentemente nessun numero maggiore di 12 può essere un divisore di 12. Pertanto: MCD(24, 12) = 12.

Dunque, in conclusione:

$$\mathrm{MCD}(36, 24) = 12.$$

ESERCIZIO. Calcola il MCD delle seguenti coppie di numeri ricorrendo all'algoritmo euclideo delle divisioni successive: (48, 64); (60, 42); (223, 25).

28.2 NOTAZIONE LINEARE STRUTTURATA

28.2.1 Un **algoritmo** si presenta dunque come ***una sequenza ordinata e finita di istruzioni, che siano effettivamente eseguibili***.

Esplicitiamo le principali proprietà di un algoritmo:
- l'algoritmo deve essere costituito da un numero finito di passi (questa proprietà è denominata ***finitezza***);
- i passi che costituiscono l'algoritmo devono essere interpretati in modo univoco dall'esecutore, uomo o macchina che sia (***non ambiguità***);
- l'esecuzione dell'algoritmo deve aver termine in un tempo finito (***terminazione***);
- l'algoritmo deve essere eseguibile effettivamente e portare ad un risultato univoco (***eseguibilità*** o ***effettività***).

A dire il vero bisognerebbe distinguere il punto di vista di chi progetta l'algoritmo da quello di chi lo esegue, specialmente nel caso in cui l'esecutore è una macchina. Ma noi non intendiamo affrontare questo problema.
Ci soffermiamo soltanto sugli schemi, espressi in forma cosiddetta *strutturata*, che costituiscono il fondamento delle regole che si ritrovano nei principali linguaggi di programmazione.
L'insieme di tali schemi è alla base di un linguaggio chiamato **Notazione Lineare Strutturata** (**NLS**), che è il linguaggio nel quale noi ci proponiamo di esprimere un algoritmo.
Le strutture, sulle quali fermeremo la nostra attenzione, sono tre ed esattamente: *sequenza*, *scelta* e *ciclo*. Ognuna di esse, salvo qualche eccezione, è sempre e comunque collocata fra le parole "Inizio" e "Fine", che di norma sottintenderemo nelle rappresentazioni che forniremo di tali schemi.

28.2.2 La struttura ***sequenza*** è costituita per l'appunto dalla sequenza di un numero n qualunque di blocchi B_1, B_2, …, B_n, che sono a loro volta istruzioni in forma strutturata. Nella figura 1 è rappresentato lo schema in NLS, mentre in figura 2 esso è espresso da un "diagramma a blocchi".

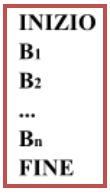

FIG. 1

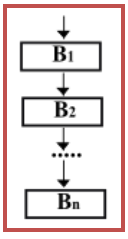

FIG. 2

Un esempio di algoritmo sequenziale è quello che risolve il problema: «calcolare il perimetro di un triangolo equilatero, nota la misura del lato». L'algoritmo risolutivo, espresso in forma retorica, è il seguente:

```
INIZIO
    leggi il valore della misura del lato
    al perimetro assegna il valore lato×3
    scrivi il valore del perimetro
FINE
```

Oppure, scritto in NLS:

```
INIZIO
    leggi(lato)
    perimetro := lato×3
    scrivi(perimetro)
FINE
```

Il simbolo ":=", come d'altro canto s'intuisce facilmente, ha come significato di assegnare alla variabile che lo precede (perimetro, nel caso specifico) il valore indicato dall'espressione che lo segue (lato×3, nell'esempio in esame).

28.2.3 La struttura ***scelta (binaria)*** valuta una "condizione" iniziale: se essa è "vera" viene eseguito successivamente il blocco strutturato di istruzioni B_1, se invece è "falsa" viene eseguito il blocco strutturato B_2. Può essere rappresentata sia in NLS (Fig. 3) sia con un diagramma a blocchi (Fig. 4).

```
SE condizione ALLORA
B1
ALTRIMENTI
B2
```

FIG. 3

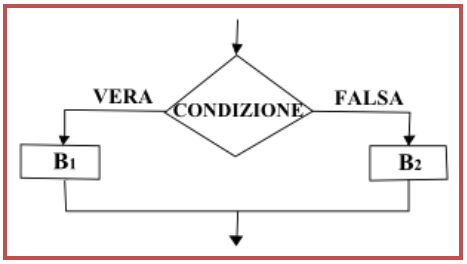

FIG. 4

Un esempio di schema siffatto è costituito dall'algoritmo che risolve il problema «se il numero intero N è dispari raddoppialo altrimenti dimezzalo». L'algoritmo risolutivo, in forma retorica, è il seguente:

```
INIZIO
    leggi il numero N
    SE N è dispari
    ALLORA
        ad N assegna il valore N×2
    ALTRIMENTI
        ad N assegna il valore N/2
FINE
```

Oppure, dopo aver stabilito che affermare che N è un numero dispari equivale a dire che N mod 2 ≠ 0, espresso in NLS:

```
INIZIO
    leggi il numero N
    M := N mod 2
    SE M ≠ 0
    ALLORA
        N := N×2
    ALTRIMENTI
        N := N/2
FINE
```

In particolare:

- se N = 35, la soluzione è 35×2 = 70;
- se N = 10, la soluzione è 10/2 = 5.

Uno dei due blocchi, mettiamo B_2, può essere anche vuoto, nel qual caso lo schema si semplifica nel modo seguente:

«SE condizione ALLORA B_1».

Prova a rappresentarlo mediante un diagramma a blocchi.

28.2.4 La struttura ***ciclo*** può essere di due tipi: uno detto *a controllo in testa* ed uno *a controllo in coda*.

- Il ***ciclo a controllo in testa*** valuta una "condizione" iniziale: se è "falsa" esce dal ciclo, se è "vera" esegue ciclicamente un blocco strutturato di istruzioni B finché la condizione rimane vera ed esce appena essa diventa falsa.

Può essere rappresentato sia in NLS (Fig. 5) sia con un diagramma a blocchi (Fig. 6).

```
MENTRE condizione
ESEGUI
B
```

FIG. 5

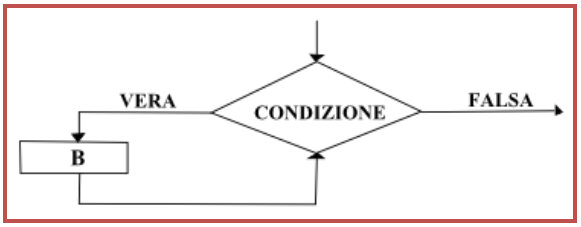

FIG. 6

Un esempio di tale costrutto è costituito dall'algoritmo che risolve il problema «mentre il valore della variabile intera A è maggiore della variabile intera B, sottrai ad A il valore di B». L'algoritmo risolutivo, espresso in forma retorica, è il seguente:

```
INIZIO
    leggi il valore di A
    leggi il valore di B
    MENTRE A è maggiore di B
    ESEGUI
        INIZIO
            calcola C = A–B
            ad A assegna il valore C
        FINE
FINE
```

Oppure, scritto in NLS:

```
INIZIO
    Leggi(A)
    Leggi(B)
    MENTRE A>B
    ESEGUI
        INIZIO
            C := A–B
            A := C
        FINE
FINE
```

In particolare, se A=27 e B=8, il ciclo è eseguito esattamente 3 volte. Si ha infatti:
27 – 8 = 19;
19 – 8 = 11;
11 – 8 = 3;
e poiché 3 non è maggiore di 8 il ciclo ha termine.

Una domanda.
Se la condizione è inizialmente falsa quante volte è eseguito il ciclo "MENTRE condizione ESEGUI B"?

- Il **ciclo a controllo in coda** esegue una prima volta un blocco strutturato B, quindi valuta una certa "condizione": se è "falsa" ripete ciclicamente il blocco B ed esce dal ciclo solo quando la condizione diventa "vera".

Anche adesso se ne può dare una rappresentazione sia in NLS (Fig. 7) sia con un diagramma a blocchi (Fig. 8).

```
RIPETI
B
FINCHE' condizione
```

FIG. 7

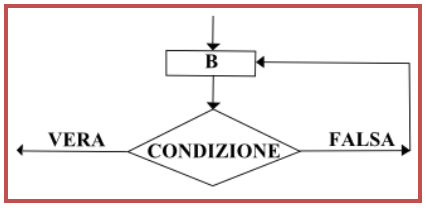

FIG. 8

Un esempio di tale costrutto è costituito dall'algoritmo che risolve il problema «aggiungi alla variabile intera A il valore M e ripeti l'operazione fino a che A diventa maggiore di B». L'algoritmo risolutivo, in forma retorica, è il seguente:

```
INIZIO
    leggi il valore di A
    leggi il valore di B
    leggi il valore di M
    RIPETI
            ad A assegna il valore A+M
    FINCHÉ A>B
FINE
```

Oppure, espresso in NLS:

```
INIZIO
    leggi(A,B,M)
    RIPETI
            A := A+M
    FINCHÉ A>B
FINE
```

La scrittura "leggi(A,B,M)" è un modo per sintetizzare le scritture "leggi(A), leggi(B), leggi(M)".
In particolare, se A=6, B=42 ed M=9, il ciclo è eseguito esattamente 5 volte, le seguenti:
6+9 = 15;
15+9 = 24;
24+9 = 33;
33+9 = 42; (il ciclo continua poiché 42 non è maggiore di 42)

42+9 = 51.

Siccome 51 è maggiore di 42, il ciclo ha termine.

Una domanda.
Se la condizione è inizialmente vera quante volte è eseguito il ciclo "RIPETI B FINCHÉ condizione"?

28.2.5 Riprendiamo l'algoritmo euclideo delle divisioni successive per scriverlo in NLS.

```
INIZIO
    leggi(a,b)
    r := a mod b
    MENTRE r≠0 ESEGUI
    INIZIO
        a := b
        b := r
        r := a mod b
    FINE
    scrivi(b)
FINE
```

Il seguente algoritmo, scritto anch'esso in NLS, risolve invece il problema di «stabilire se un dato numero naturale dispari (maggiore di 1) è primo». Ricordiamo che l'operazione "A div B" calcola il divisore di A per B.

```
INIZIO
    leggi(numero)
    divisore := 1
    RIPETI
        divisore := divisore + 2
        quoziente := numero div divisore
        resto := numero mod divisore
    FINCHÉ resto = 0 o quoziente < divisore
    SE resto = 0 ALLORA
        il numero non è primo
    ALTRIMENTI
        il numero è primo
FINE
```

Puoi costatare che fra le parole "Ripeti" e "Finché" non abbiamo scritto le istruzioni "Inizio" e "Fine". Non si tratta di un errore dal momento che le istruzioni interne al ciclo "Ripeti … Finché" costituiscono una sequenza che rende superflue le parole "Inizio" e "Fine". Se comunque tali parole si mettono, va bene lo stesso.

28.2.6 Ancora un riferimento ad Euclide. Al giorno d'oggi calcoliamo il minimo comune multiplo di due numeri (mcm), fattorizzando i numeri e prendendo i fattori comuni e non comuni con il maggiore esponente.

Per esempio, se i numeri sono 36 e 30, dapprima si trova che: $36=2^2\cdot3^2$ e $30=2\cdot3\cdot5$. Si ha quindi: $\text{mcm}(36,30) = 2^2\cdot3^2\cdot5 = 180$.

Euclide però aveva fornito un procedimento diverso negli *Elementi* (libro VII, prop. 34), in base al quale mcm(a,b) = ab/MCD(a,b), vale a dire che il mcm di due numeri a, b si ottiene dividendo il prodotto dei due numeri per il loro MCD.
Nel caso dell'esempio prima considerato, una volta costatato che $\mathrm{MCD}(36,30)=6$, con il procedimento di Euclide si ha: $\mathrm{mcm}(36,30) = 36\times30/6 = 180$. Come sopra ovviamente.

Ti proponiamo per esercizio di scrivere un algoritmo che risolva il problema di determinare il minimo comune multiplo di due numeri con il procedimento di Euclide.

28.2.7 Avere scritto in forma strutturata un insieme di istruzioni non significa necessariamente che si abbia a che fare con un algoritmo. In effetti le istruzioni, ad esempio, possono dar luogo ad una situazione in cui di fatto il procedimento di calcolo non ha fine, come invece dovrebbe essere se si trattasse di un algoritmo. Un contro-esempio è sufficiente per far capire cosa intendiamo dire.
Sia allora il seguente schema di calcolo:

```
INIZIO
    X := 6
    MENTRE X<20 ESEGUI
        X := (X–3)×2
    scrivi(X)
FINE
```

Il valore di X, ad ogni ciclo, è 6, per cui la condizione X<20 è sempre vera e pertanto il ciclo si ripete all'infinito. Si dice che il procedimento entra in un "loop".
In questo caso non si può parlare di algoritmo.

28.3 FUNZIONI CALCOLABILI

28.3.1 Fin dai tempi antichi, oltre 2000 anni prima di Cristo, con le civiltà egizia e babilonese, operazioni come l'addizione e la moltiplicazione erano considerate effettivamente calcolabili. Detto in un linguaggio più appropriato, considerati due qualsiasi numeri naturali x ed y, le funzioni di due variabili f(x,y)=x+y e g(x,y)=xy hanno valori definiti per ogni coppia ordinata (x,y). Ad esempio:

$$f(2,3) = 2+3 = 5,\quad g(2,3) = 2\times3 = 6.$$

Ora, se il concetto di "calcolabilità" può anche sembrare chiaro sotto l'aspetto intuitivo, non lo è affatto dal punto di vista della formalizzazione. In effetti, una definizione formale di "funzione calcolabile" (o "computabile") non è una cosa semplice e immediata. Al nostro livello di studi dobbiamo accontentarci di una definizione che, senza essere una definizione rigorosa, ne dia tuttavia un'idea abbastanza sicura. Ebbene, con questo limite, accettiamo la seguente definizione:

Una funzione si dice ***calcolabile*** se esiste un procedimento di calcolo automatico, vale a dire un algoritmo, capace di ricavarne i valori in corrispondenza dei valori attribuiti alle variabili indipendenti.

In questo senso le due funzioni sopraddette, f(x,y) e g(x,y), sono *funzioni calcolabili*.

È calcolabile anche la seguente funzione:

$$f(n)=\begin{cases}0 & \text{se n è un numero naturale pari}\\ 1 & \text{se n è un numero naturale dispari}\end{cases}$$

28.3.2 Sono funzioni calcolabili le cosiddette *funzioni ricorsive*. Incominciamo con la definizione:

Una **funzione ricorsiva** (o **definita ricorsivamente**) è una funzione che richiama se stessa nel calcolo dei vari termini.

Il primo esempio noto di funzione definita ricorsivamente in modo esplicito è costituito dalla funzione che genera i cosiddetti numeri di Fibonacci [(1)]. Essi formano la seguente successione:

$$\mathbf{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \dots , f_{n-2}+f_{n-1}, \dots ,}$$

nella quale dopo i primi due numeri, entrambi uguali ad 1, ogni altro numero di posto n è la somma dei due immediatamente precedenti, cioè di quelli di posto $n-2$ ed $n-1$.

Tale successione di numeri è il risultato del seguente problema, qui proposto in italiano ma enunciato da Fibonacci in lingua latina:

«*Un tale pose una coppia di conigli in un luogo circondato del tutto da pareti, per scoprire quante coppie nascessero da quella in un anno: com'è nella loro natura, ogni coppia di conigli genera ogni mese un'altra coppia e comincia a procreare a partire dal secondo mese dalla nascita*. *(La coppia iniziale comincia a riprodursi a partire dalla fine del primo mese). Quante coppie di conigli discendono in un anno dalla coppia (originaria)?*»

La soluzione, che peraltro lo stesso Fibonacci fornisce facendo seguito alla traccia del problema, non è difficile e, almeno per i primi 5 mesi è visualizzata nel grafo sottostante (Fig. 9).

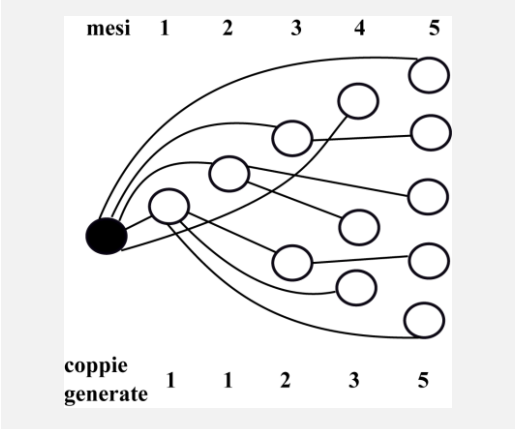

FIG. 9

Naturalmente per rispondere al quesito bisogna considerare i 12 mesi di un anno, per cui si ha la seguente successione:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144;$$

la somma di questi numeri, che è 376, dà il numero di coppie di conigli discendenti da una coppia in un anno. Se, però, si vuole annoverare pure la coppia originaria, è evidente che il numero complessivo delle coppie è 377.

Ti proponiamo di risolvere il seguente esercizio:
Quante coppie di conigli sono generate nei primi cinque mesi del secondo anno se tutte le coppie esistenti nel primo anno continuano a figliare?

La funzione che genera la successione di Fibonacci è la seguente:

[1] **Fibonacci**, Leonardo, detto anche Pisano, 1175-1235.

$$f(n)=\begin{cases}1 \text{ se } n=1 \text{ oppure } n=2\\ f(n-2)+f(n-1) \text{ se } n>2\end{cases}$$

ed è evidente che si tratta di una funzione che richiama se stessa nella determinazione dei vari numeri dopo il secondo.

Un algoritmo che generi i numeri di Fibonacci (minori di 400) è il seguente:

```
inizio
    a := 1
    b := 1
    c := a+b
    mentre c < 400 esegui
        inizio
        a := b
        b := c
        c := a+b
    fine
fine
```

28.3.3 Vediamo adesso qualche altro semplice esempio di funzione ricorsiva.

- ESEMPIO 1. **Somma dei primi n numeri naturali a partire da 1**.

La somma f(n) dei primi n numeri naturali a partire da 1 è evidentemente una funzione di n. Precisamente:

$$f(n) = 1 + 2 + 3 + 4 + \ldots + n.$$

Una volta constatato che $f(1)=1$, è abbastanza semplice capire che risulta:

- $f(2) = 1+2 = f(1)+2$,
- $f(3) = f(2)+ 3$,
- $f(4) = f(3)+4$,

e così via.

In generale, se $n>1$, si ha $f(n)=f(n-1)+n$. Per cui la forma ricorsiva della funzione f(n) è la seguente:

$$f(n)=\begin{cases}1 \text{ se } n=1\\ f(n-1)+n \text{ se } n>1\end{cases}$$

Prova a scrivere l'algoritmo che risolve il problema di determinare la somma dei primi 100 numeri naturali a partire da 1.

- ESEMPIO 2. **Successione dei primi n numeri naturali non nulli**.

La successione dei numeri naturali non nulli è evidentemente:

$$1, 2, 3, 4, \ldots, n.$$

Per cui, constatato che l'ennesimo termine f(n) della successione è $f(n)=n$, risulta:

- $f(1) = 1$,
- $f(2) = 1+1 = f(1)+1$,
- $f(3) = f(2)+1$,

e così via.

In generale, se $n>1$, si ha: $f(n)=f(n-1)+1$. Pertanto la forma ricorsiva della funzione è la seguente:

$$f(n)=\begin{cases}1 \text{ se } n=1\\ f(n-1)+1 \text{ se } n>1\end{cases}$$

Prova a scrivere l'algoritmo che genera i primi 50 numeri naturali non nulli.

Altri esempi te li proponiamo per esercizio:

Scrivere la forma ricorsiva delle seguenti funzioni f(n):

a) f(n) è la somma dei primi n numeri naturali pari a partire da 2.
b) f(n) è la somma dei primi n numeri naturali dispari a partire da 1.
c) f(n) è la somma dei quadrati dei primi n numeri naturali a partire da 1.
d) f(n) è la somma dei cubi dei primi n numeri naturali a partire da 1.

28.3.4 Bisogna sapere che ***non tutte le funzioni sono calcolabili***. Per dimostrarlo basterebbe fornire un contro-esempio di funzione non calcolabile. Ebbene, un contro-esempio siffatto, quantunque banale, è costituito dalla funzione che elenca tutti gli elementi di un insieme numerico infinito (numeri naturali, numeri interi, numeri razionali compresi fra due interi distinti, eccetera). Questo per il fatto ovvio che nessun algoritmo è in grado di enumerare tutti gli elementi di un insieme infinito.

Esistono comunque altri esempi più sofisticati, ma non ce ne possiamo occupare al nostro livello di studi, troppo elementare. Ciò non di meno, vogliamo soffermarci ugualmente su un esempio particolarmente interessante. Ma per questo è necessaria una breve premessa.

È facile constatare che l'equazione:

$$x+y=z$$

è soddisfatta da almeno una terna di numeri interi positivi (in realtà tali terne sono infinite).

È noto altresì che anche l'equazione:

$$x^2+y^2=z^2$$

è soddisfatta da almeno una terna di numeri interi positivi (pure in questo caso tali terne sono infinite: sono le cosiddette terne pitagoriche).

Nel tentativo di generalizzare la questione, il francese Pierre de Fermat (1601-1665) si occupò dell'equazione:

$$x^n+y^n=z^n$$

con n>2. Formulò la congettura che, in questo caso, non esiste alcuna terna di numeri interi positivi che soddisfi l'equazione.

Questa congettura rimase tale per oltre tre secoli, anche se i matematici riuscirono a dimostrare che era vera in molti casi (in particolare per n=3 e per n=5). Finché nel 1994 il matematico britannico Andrew Wiles (n. 1953) non ne fornì una dimostrazione rigorosa, che per la verità aveva annunciato l'anno prima.

Ciò detto, consideriamo la seguente funzione:

$$f(n)=\begin{cases} 0 & \text{se esiste una terna di numeri interi positivi } (x,y,z) \text{ tali che } x^n+y^n=z^n \\ 1 & \text{se non esiste alcuna terna siffatta} \end{cases}$$

Ebbene, fino alla scoperta di Wiles si poteva dire che $f(1)=f(2)=0$ ed anche $f(3)=f(5)=1$, ma non si poteva dire alcunché per un generico $n>2$. Per questo motivo la funzione f(n) doveva essere considerata *non calcolabile*.

Invece, dopo la scoperta di Wiles, si può affermate che:

$$f(n)=0 \text{ per } n=1 \text{ e per } n=2, \text{ mentre } f(n)=1 \text{ per ogni } n>2.$$

Ragion per cui la funzione è da considerarsi *calcolabile*.

VERIFICHE

1. Sulla base del seguente algoritmo completa la tabella sottostante:

```
inizio
    leggi(a,b)
    a := a·a
    b := b·b
    c := a+b
    scrivi(c)
fine
```

a	0	1	1	3
b	0	1	2	4
c				

2. Sulla base del seguente algoritmo completa la tabella sottostante:

```
inizio
    leggi(a,b)
    a := 2·a
    b := b·b
    c := a·b
    scrivi(c)
fine
```

a	0	2	3	2
b	1	2	2	3
c				

3. Sulla base del seguente algoritmo completa la tabella sottostante:

```
inizio
    leggi(n)
    se n>5 allora
        a := n–5
    altrimenti
        a := n·n
    scrivi(a)
fine.
```

n	3	5	7	10
a				

4. Stabilire che differenza c'è tra i due seguenti algoritmi, dopo aver trovato i risultati in uscita:

```
inizio
    a := 3
    b := 4
    a := b
    b := a
    scrivi(a,b)
```

```
inizio
    a := 3
    b := 4
    c := a
    a := b
    b := c
```

```
    fine                                        scrivi(a,b)
                                              fine
```

5. Considera il seguente algoritmo:

```
inizio
a := 0
mentre a < 100 esegui
    inizio
        scrivi(a)
        a := a+2
    fine
fine
```

Quale problema risolve questo algoritmo? Individua l'unica alternativa corretta tra le seguenti.

[A] Permette di riconoscere se un numero è minore di 100;

[B] Permette di stabilire se un numero minore di 100 è pari;

[C] Genera i numeri pari minori di 100;

[D] Calcola la somma dei numeri pari minori di 100.

6. Che cosa calcola il seguente algoritmo, tenendo presente che n è un naturale non nullo?

```
    inizio
        leggi(n)
        a := 1
        p := 1
        mentre a<n esegui
            inizio
                a := a+1
                p := p·a
            fine
        scrivi(p)
    fine
```

7. Che cosa calcola il seguente algoritmo?

```
    inizio
        leggi(a,b)
        x := a
        y := b
        mentre x≠y esegui
            se x<y allora
                x := x+a
            altrimenti
                y := y+b
        scrivi(x)
    fine
```

8. Considera il seguente algoritmo, conosciuto come ***algoritmo di Siracusa***:

```
    inizio
        leggi(N)
        mentre N>1 esegui
```

```
            se N è pari allora
                N:=N div 2
            altrimenti
                N:=3N+1
        scrivi(N)
    fine
```

Prova a calcolare la sua uscita per i seguenti valori di N: 10, 17, 15. Noterai una circostanza interessante: il risultato è sempre 1. Ebbene pare che questo accada per ogni numero N, ma finora nessuno è stato in grado di dimostrarlo, anche se il fatto è stato verificato per numeri fino a 20×2^{58}. È stata solamente formulata la congettura che sia così, vale a dire: *qualunque sia il numero di partenza, intero maggiore di 1, l'operazione ripetuta porta sempre, dopo un numero finito di passi, al numero 1*. La congettura è stata formulata nel 1937 dal matematico tedesco Lothar Collatz (1910-1990) ed è per questo conosciuta come ***congettura di Collatz***.

9. Scrivi in NLS gli algoritmi che risolvano i problemi seguenti:
- a) Determinare se un dato numero naturale n è pari oppure dispari.
- b) Dato un numero naturale n non nullo, determinare tutti i suoi divisori.
- c) Di ogni numero naturale minore di 15 scrivere il successivo se il numero è pari e il suo quadrato se è dispari.
- d) Dati due numeri naturali a e b diversi da 0, calcolare il loro MCD ed il loro mcm.
- e) Calcolare la somma dei primi n numeri naturali, a partire da 1.
- f) Calcolare il prodotto dei primi n numeri naturali, a partire da 1.
- g) Scrivere la successione dei multipli di a non maggiori di b.
- h) Dati i due numeri naturali n e k, scrivere i primi n multipli di k.

10. Si consideri la funzione f(n), dove $n\in\mathbb{N}_0$, la quale genera numeri interi. Scrivere la funzione in forma ricorsiva, sapendo che:

a) $f(n) = 3n+1$, b) $f(n) = n(n+1)$, c) $f(n) = 2\cdot3^n$.

11. Definire per ricorsione la funzione $f(n)=a^n$, dove $n\in\mathbb{N}_0$ ed a è un qualsiasi numero reale non nullo (si tratta evidentemente della definizione ricorsiva della **potenza di un numero con esponente naturale**).

UNA BREVE SINTESI PER DOMANDE E RISPOSTE

DOMANDE.

1. Come si può definire, ancorché in modo non formalmente rigoroso, un algoritmo?
2. Quali sono le strutture fondamentali nella Notazione Lineare Strutturata?
3. Se la condizione del costrutto "RIPETI B FINCHÉ condizione" è inizialmente vera, quante volte è eseguito il ciclo?
4. Scrivi in NLS un algoritmo idoneo a determinare tutti i divisori di 60.
5. Il massimo comune divisore di tre numeri assegnati è 1. Si può concludere che il loro minimo comune multiplo è dato dal loro prodotto?
6. Come si può definire, ancorché in modo non formalmente rigoroso, una funzione calcolabile?

7. Cos'è una funzione ricorsiva?
8. Scrivere la forma ricorsiva della funzione f(n) uguale alla somma dei cubi dei primi n numeri naturali a partire da 1.

RISPOSTE.

1. Un algoritmo si può definire come una sequenza ordinata e finita di istruzioni, che siano effettivamente eseguibili.
2. Sono la "sequenza", la "scelta (binaria)" ed il "ciclo".
3. Si tratta del ciclo a controllo in coda e siccome si ripete ogni volta che la condizione è falsa, mentre termina appena la condizione è vera, si capisce che se la condizione iniziale è vera, il procedimento ha termine senza che venga eseguito alcun ciclo.
4. Un algoritmo idoneo a determinare tutti i divisori di 60 è il seguente:

```
INIZIO
    N := 60
    RIPETI
        Divisore := 60 div N
        Resto := 60 mod N
        SE Resto=0 scrivi(Divisore)
        N := N–1
    FINCHÉ N<1
FINE
```

5. No, non è così. Basti pensare ai tre numeri 2, 3, 4, il cui massimo comune divisore è 1 ed il cui minimo comune multiplo è 12, ma il loro prodotto è 24.
6. Una funzione si dice calcolabile se esiste un procedimento di calcolo automatico, vale a dire un algoritmo, capace di ricavarne i valori in corrispondenza dei valori attribuiti alle variabili indipendenti
7. Si definisce funzione ricorsiva una funzione che richiama se stessa nella costruzione dei vari termini.
8. Constatato che si ha: $f(n) = 1^3+2^3+3^3+...+n^3$, è facile comprendere che la forma ricorsiva della funzione è $f(n)=\begin{cases} 1 \text{ se } n=1 \\ f(n-1)+n^3 \text{ se } n>1 \end{cases}$