

Algoritmi e diagrammi di flusso

Alessandro Fuser

6 marzo 2019

Indice

| | | |
|----------|-------------------------------------|----------|
| 1 | Teoria | 3 |
| 1.1 | Gli algoritmi | 3 |
| 1.1.1 | Proprietà degli algoritmi | 3 |
| 1.1.2 | Esempi di algoritmo | 4 |
| 1.1.3 | Sequenza | 5 |
| 1.1.4 | Selezione | 5 |
| 1.1.5 | Iterazione | 6 |
| 1.2 | Diagrammi di flusso | 7 |
| 1.2.1 | Sequenza | 7 |
| 1.2.2 | Selezione | 8 |
| 1.2.3 | Iterazione | 8 |
| 1.2.4 | Esempi | 8 |
| 2 | Esercizi | 9 |
| 2.1 | Struttura di selezione | 9 |
| 2.2 | Struttura di iterazione | 10 |
| 2.3 | Riassuntivi | 10 |

1 Teoria

1.1 Gli algoritmi

Un **algoritmo** è la descrizione del percorso risolutivo di un problema per giungere dai dati iniziali ai risultati finali. Questo viene scritto pensando di rivolgersi a un **esecutore**, capace di svolgere azioni descritte da **istruzioni**, scritte in un particolare **linguaggio**.

Un'altra possibile definizione di algoritmo è la seguente: "Un algoritmo è un procedimento di calcolo che si basa sull'applicazione di un numero finito di regole che determinano in modo meccanico TUTTI i singoli passi del procedimento stesso". In parole più brevi, potremmo definirlo come ogni procedimento che consente di risolvere un problema. L'algoritmo è un concetto fondamentale dell'informatica, anzitutto perché è alla base della nozione teorica di calcolabilità, in quanto un **problema è calcolabile solo quando è risolvibile mediante un algoritmo**. Se ci pensate, usiamo algoritmi tutti i giorni per risolvere ogni tipo di problema che ci si presenta davanti.

Le principali operazioni presenti all'interno di un algoritmo sono:

| Istruzioni di | Per |
|---------------------|---|
| Input | Ricevere dati (numeri, espressioni, testi) |
| Output | Mandare messaggi comunicare risultati |
| Assegnazione | Memorizzare un dato associandolo al nome di una variabile |
| Calcolo | Svolgere operazioni tra i dati |

Quando mi serve salvare un qualche valore o dato, posso fare uso di una variabile. Una **variabile** è un contenitore di dati destinato a contenere dei valori, che può essere modificata durante l'esecuzione di un programma. Ad esempio, se voglio salvare l'informazione che un lato di un rettangolo ha valore 10, allora mi basta assegnare tale valore ad una variabile, che avrà un proprio nome, del tipo:

```
cateto = 10;
base = 15;
altezza = 5;
```

Certe variabili non vengono assegnate direttamente dall'utente, ma vengono prodotte nell'esecuzione di un programma. Pensa ad esempio al calcolo dell'area di un rettangolo. Per fare tale operazione, ti servirà il valore della base e dell'altezza del rettangolo (che salverai in apposite variabili) e, per salvare il valore dell'area, userai una nuova variabile, del tipo:

```
area = base * altezza;
perimetro = (base + altezza) * 2;
```

Quando vuoi salvare un valore per poterlo usare successivamente all'interno di un programma, salvalo in una variabile, di cui decidi tu il nome.

I tipi principali che una variabile può assumere sono:

| Tipo | Per |
|---------------|------------------------------------|
| int | Numeri interi (0, 1, 3, ...) |
| double | Numeri con la virgola o frazionari |
| string | Parole o lettere |
| bool | Valori vero o falso (1 e 0) |

1.1.1 Proprietà degli algoritmi

Le proprietà principali di un algoritmo sono 5:

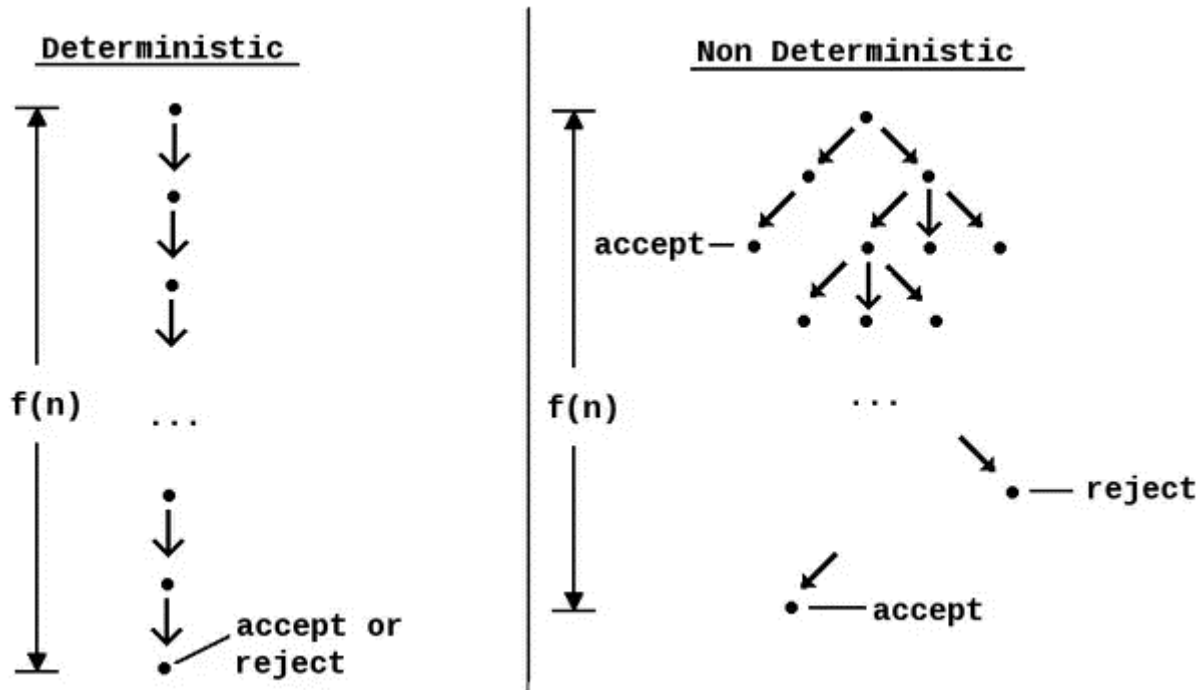
| Nome Proprietà | Descrizione |
|----------------|--|
| Generale | Deve risolvere una classe di problemi e non un singolo problema |
| Finito | Devo avere un numero ben preciso di passi da svolgere |
| Completo | Deve tenere conto di tutti i possibili casi da risolvere |
| Non ambiguo | Ogni istruzione deve essere definita in modo preciso ed univoco |
| Eseguibile | Deve esistere qualcuno o qualcosa che sia in grado di eseguire l'algoritmo |

Altre proprietà molto utili sono:

- **Modularità**, ossia la capacità di suddividere il problema generale in sotto-problemi più semplici;
- **Gerarchia**, ossia l'esistenza di un ordine ben specifico in cui le operazioni vengono svolte;

- **Realizzabilità**, ossia la capacità, da parte di chi legge l'algoritmo, di poterlo eseguire con quello che ha a disposizione;
- **Efficienza**, ossia la diversa velocità di esecuzione degli algoritmi a parità di compito svolto

Le tipologie di algoritmo, invece, possono essere principalmente due: **deterministico** se, per ogni istruzione, esiste un solo passo successivo possibile; **non deterministico** se contiene almeno un'istruzione che ammette più passi successivi. Una volta che verranno spiegate le strutture dell'algoritmo, questo concetto diverrà molto più chiaro ed immediato.



1.1.2 Esempi di algoritmo

Esempi di algoritmi scritti in linguaggio comune di tutti i giorni possono essere i seguenti: calcolo del perimetro di un rettangolo, misura di un cateto, massimo fra quattro numeri.

Vediamo la risoluzione del **calcolo del perimetro di un rettangolo**:

1. Inizia
2. Acquisisci il valore della base B e dell'altezza H
3. Somma i valori di B e H ed assegnalo a P
4. Raddoppia il valore di P ed assegnalo a 2P
5. Rendi noto il valore di P
6. Fine

Risoluzione della **misura del cateto**:

1. Inizio
2. Acquisisci l'ipotenusa A e il cateto B
3. Controlla se $A \geq B$
4. Se è vero, calcola $\sqrt{A^2 - B^2}$ ed assegna il valore a C e rendi noto C
5. Se è falso, scrivi un messaggio di errore
6. Fine

Risoluzione del **massimo fra quattro numeri**:

1. Inizio
2. Leggi il primo numero e memorizzalo nella variabile M

3. Ripeti la seguente operazione per 3 volte: leggi un numero e memorizzalo nella variabile N e se N è maggiore di M allora assegna il valore di N a M
4. Rendi noto M
5. Fine

Esempio su un indovinello famoso, quello del **trasporto della capra, del lupo e del cavolo**: Un pastore ha una pecora, un lupo e un cavolo. Deve portarli dall'altra parte di un fiume che non ha né ponti né guadi, facendo ben attenzione che nel tragitto la pecora non sbafi il cavolo o - peggio - il lupo non sbrani la pecora. Il pastore ha a disposizione una piccola canoa che può contenere solo una cosa o un animale oltre al pastore. Quindi non è possibile fare il trasbordo in un'unica soluzione. Non ha nessun problema a far avanti e indietro nel fiume, ma si deve, invece, fare attenzione anche a cosa combinano i due rimasti soli sulla riva mentre il pastore è in acqua con il terzo. Come fa il pastore a trasportare pecora, cavolo e lupo, tutti e tre sani, salvi e... non mangiati?

1. Porta la capra sull'altra sponda
2. Torna indietro da solo
3. Porta il cavolo dall'altra parte e carica la pecora
4. Porta la capra indietro
5. Scarica la capra e porta il lupo sull'altra sponda
6. Torna indietro da solo
7. Porta la capra sull'altra sponda

1.1.3 Sequenza

Purtroppo, le macchine (computer) che non capiscono il linguaggio naturale usato dagli esseri umani, per cui bisogna usare un linguaggio più definito e conciso, senza possibilità di ambiguità.

Usano un linguaggio più appropriato, dunque, possiamo risolvere il problema del calcolo del perimetro di un rettangolo nel seguente modo:

```

Inizio
SCRIVI "Fornisci la misura della base B"
LEGGI B
SCRIVI "Fornisci la misura dell'altezza H"
LEGGI H
P:= B+H
2P:= 2*P
SCRIVI "La misura del perimetro: "
SCRIVI P
Fine

```

Notiamo i seguenti fatti, che **valgono come regola generale** quando si scrive un algoritmo:

- Con il simbolo $:=$ si indica l'assegnazione di un valore ad una variabile
- Con le istruzioni del tipo SCRIVI chiediamo all'esecutore di rivolgersi all'utente, comunicando un risultato
- Con le istruzioni del tipo LEGGI chiediamo all'esecutore di rivolgersi all'utente chiedendo un valore

In questo algoritmo le istruzioni sono da eseguire sempre e una sola volta, nell'ordine esatto in cui si presentano. Una struttura come questa viene chiamata **SEQUENZA**.

1.1.4 Selezione

Molte volte, nella risoluzione di un problema, siamo posti davanti a delle scelte, per cui, in base a quella che faccio, devo fare determinate operazioni. Vediamo un esempio, per la misura del cateto vista precedentemente:

```

Inizio
SCRIVI "Dammi la misura dell'ipotenusa"
LEGGI A
SCRIVI "Dammi la misura del cateto"
LEGGI B

```

```

SE (A>B)
    ALLORA
        C:=  $\sqrt{A^2 - B^2}$ 
        SCRIVI "L'altro cateto misura"
        SCRIVI C
    ALTRIMENTI
        SCRIVI "Errore"
Fine

```

In questo algoritmo, l'istruzione $C := \sqrt{A^2 - B^2}$ viene eseguita **soltanto se** la condizione $A \leq B$ è vera, l'istruzione SCRIVI "Errore" solo se la condizione è falsa. Una struttura di questo tipo viene chiamata **SELEZIONE**.

1.1.5 Iterazione

Altre volte, in certi problemi, dobbiamo ripetere una determinata operazione fino a che una certa condizione non diventa vera. Pensa ad un muratore, che, per costruire un muro, deve impilare mattoni fino a che non raggiunge una certa altezza. Lo stesso possiamo farlo con un linguaggio più formale, come nel caso del **determinare il massimo tra 4 numeri**:

```

Inizio
SCRIVI "Fornisci il primo numero"
LEGGI M
CONT := 1
RIPETI
    CONT := CONT + 1
    SCRIVI "Fornisci un altro numero"
    LEGGI N
    SE N > M
        ALLORA
            M := N
FINCHE' CONT < 4
SCRIVI "Il massimo risulta"
SCRIVI M
Fine

```

Facciamo notare le seguenti cose:

- **RIPETI istruzioni FINCHE' condizione** continua a far eseguire le istruzioni fintanto che la condizione è vera; quando la condizione diventa falsa, l'esecutore prosegue con l'istruzione alla riga successiva del FINCHE'
- **CONT** ha la funzione di variabile contatore, in modo da tenere traccia di quante volte sto facendo l'operazione

Un altro modo per scrivere il ciclo potrebbe essere:

```

Inizio
SCRIVI "Fornisci il primo numero"
LEGGI M
CONT := 1
PER i CHE VA DA 1 A 3 CON PASSO 1 FAI
    CONT := CONT + 1
    SCRIVI "Fornisci un altro numero"
    LEGGI N
    SE N > M
        ALLORA
            M := N
PROSSIMO i
SCRIVI "Il massimo risulta"
SCRIVI M
Fine

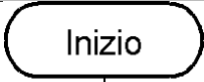
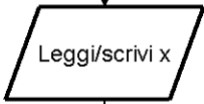
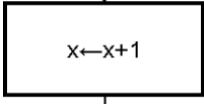
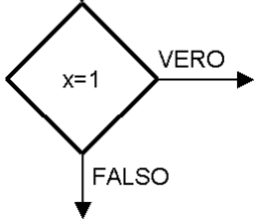
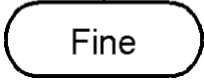
```

In questo modo, specifichiamo per bene e in maniera non ambigua il nome della variabile, il suo valore di inizio e di fine e di quanto deve aumentare alla fine di ogni ciclo.

Questa struttura di ripetere le stesse operazioni per un tot di volte prende il nome di **ITERAZIONE**.

1.2 Diagrammi di flusso

Un'alternativa al linguaggio strutturato e pragmatico dell'algoritmo, si può utilizzare un modo grafico detto **diagramma a blocchi**. Elenco da subito le forme convenzionali dei blocchi:

| Immagine | Descrizione |
|---|-------------|
|  | |
|  | |
|  | |
|  | |
|  | |

Le tre strutture fondamentali degli algoritmi possono essere descritte così con i diagrammi a blocchi. I blocchi vengono rappresentati uniti da frecce, che indicano il flusso di esecuzione dell'algoritmo.

1.2.1 Sequenza

I blocchi di apertura e chiusura dell'algoritmo hanno una sola sola freccia, rispettivamente uscente ed entrante, mentre i blocchi intermedi hanno generalmente una freccia entrante e una uscente



Figura 1: Struttura generale della sequenza

1.2.2 Selezione

Il blocco di controllo, nel caso più semplice, ha una freccia entrante e due uscenti, corrispondenti ai due valori di verità della condizione che viene valutata.

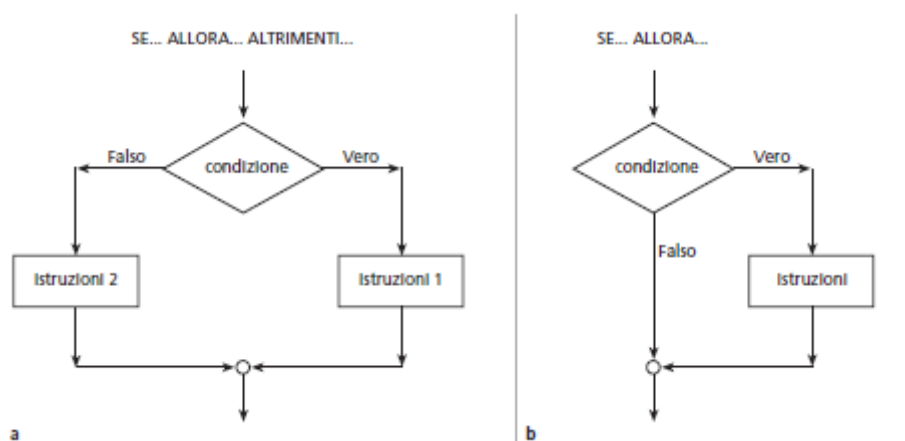


Figura 2: Struttura generale della selezione

1.2.3 Iterazione

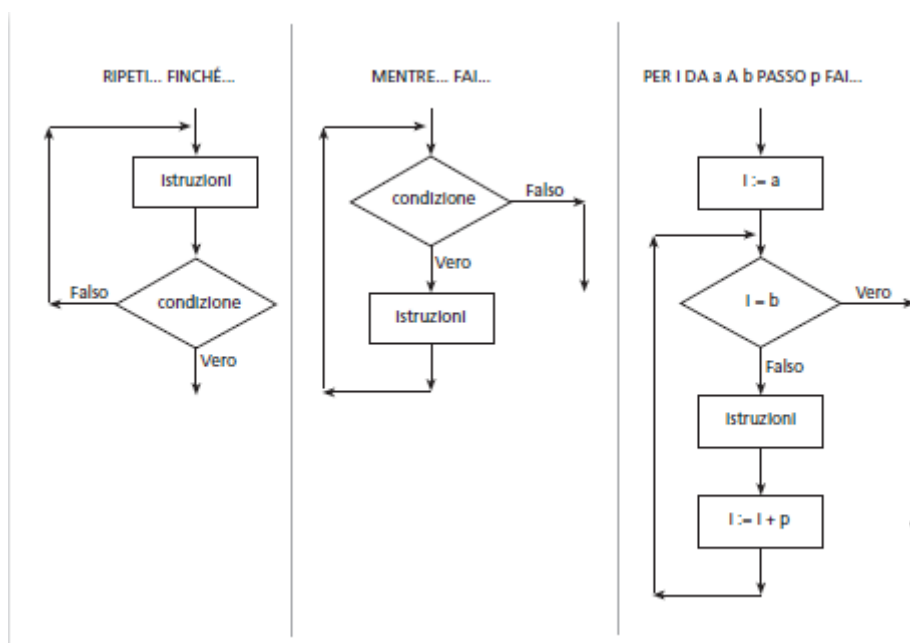


Figura 3: Struttura generale dell'iterazione

1.2.4 Esempi

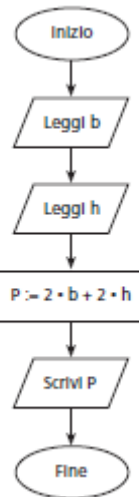


Figura 4: Calcolo del perimetro di un rettangolo

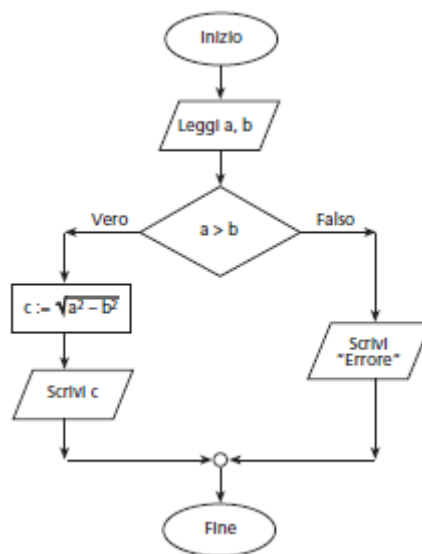


Figura 5: Calcolo del cateto di un triangolo rettangolo

2 Esercizi

2.1 Struttura di selezione

1. Dati due numeri interi positivi N1 e N2, calcolare e visualizzare il numero massimo MAX tra i due
2. Dati tre numeri interi positivi N1, N2, N3, calcolare e visualizzare il numero massimo MAX tra i tre
3. Dati due numeri interi positivi N1 e N2, calcolare e visualizzare il numero minimo MIN tra i due
4. Dati tre numeri interi positivi N1, N2, N3, calcolare e visualizzare il numero minimo MIN tra i tre
5. Dato un numero intero positivo N, verificare se appartiene all'intervallo [4,10]
6. Dato un numero intero positivo N, verificare se appartiene all'intervallo [4,10] o all'intervallo [15,20]
7. Dati 5 numeri N1, N2, N3, N4, N5, calcolare e visualizzare la somma SUM dei numeri pari
8. Dati 5 numeri N1, N2, N3, N4, N5, calcolare e visualizzare il prodotto dei numeri multipli di 3
9. Dato un numero N, dire se è un multiplo di 7
10. Dati due numeri N1, N2, dire se N1 è un multiplo di N2 o se N2 è un multiplo di N1

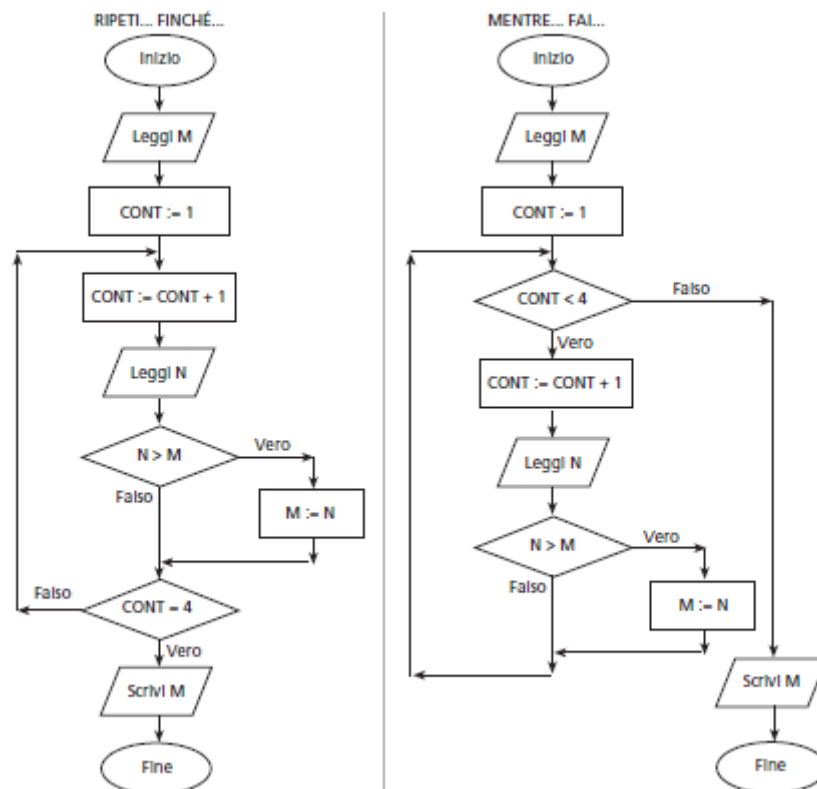


Figura 6: Calcolo del massimo tra 4 numeri

2.2 Struttura di iterazione

1. Dato un numero N, generare e visualizzare il numero successivo
2. Dato un numero N, generare e visualizzare in ordine crescente i numeri minori di N
3. Dato un numero N, generare e visualizzare in ordine crescente i numeri dispari minori o uguali di N
4. Dato un numero N, generare e visualizzare in ordine crescente i numeri pari minori o uguali di N
5. Dati due numeri N1 e N2, generare e visualizzare in ordine crescente i numeri compresi tra N1 e N2 (o viceversa)
6. Dato un numero N, generare e visualizzare in ordine decrescente i numeri minori di N
7. Dati due numeri N1 e N2, generare e visualizzare in ordine decrescente i numeri compresi tra N1 e N2 (o viceversa)
8. Dato un numero N, calcolare e visualizzare la somma dei numeri minori di N
9. Dato un numero N, calcolare e visualizzare la somma dei numeri dispari minori di N
10. Dati due numeri N1, N2, calcolare, mediante somma ripetuta, e visualizzare il prodotto dei due numeri
11. Dato un numero N, calcolare e visualizzare il prodotto dei numeri minori di N
12. Dati due numeri B,E, calcolare e visualizzare la potenza in base B ed esponente E, mediante il metodo delle moltiplicazioni ripetute
13. Dato un numero N, verificare se N è un numero primo
14. Dato un numero N, calcolare e visualizzare tutte le coppie di numeri che danno per somma il numero stesso
15. Dato un numero N, calcolare e visualizzare tutte le coppie di numeri che danno per prodotto il numero stesso

2.3 Riassuntivi

1. Dato un numero N, calcolare il numero successivo del doppio del quadrato di N
2. Dato un numero N, calcolare il quadrato del successivo del doppio del numero N
3. Dato un numero N, calcolare la differenza tra il quadrato del numero ed il numero stesso

4. Dati due numeri N_1 , N_2 , verificare se uno dei è il quadrato dell'altro
5. Dati quattro numeri N_1 , N_2 , N_3 , N_4 , verificare se costituiscono la proporzione $N_1:N_2=N_3:N_4$
6. Data l'area A e la base B di un rettangolo, calcolarne il perimetro P
7. Dato il perimetro P e l'altezza H di un rettangolo, calcolare l'area A
8. Data la diagonale D ed un lato C di un rettangolo, calcolare l'area A del quadrato costruito sul lato maggiore
9. Dati tre numeri A , B , C , determinare se A può essere l'ipotenusa di un triangolo rettangolo con i cateti uguali a B e C
10. Data la misura di un angolo, verificare se è acuto, ottuso o retto
11. Data una misura di tempo espressa in minuti M_1 , convertirla in ore H e minuti M
12. Data una misura di tempo espressa in secondi S_1 , convertirla in ore H , minuti M e secondi S
13. Date due misure espresse in ore e minuti, calcolare la differenza fra la prima e la seconda
14. A fronte di un pagamento con una banconota da 100euro, calcolare il resto da fornire, usando solo banconote e monete esistenti
15. Si conosce il prezzo di un oggetto O comprensivo di IVA (22%), determina il prezzo netto
16. Dato il prezzo P , se tale importo è maggiore di 300 allora fare uno sconto del 5%
17. Dato un numero N , trasforma e visualizza il corrispondente codice binario
18. Dato il numero di scatti telefonici N e sapendo che la bolletta del telefono viene calcolata nel seguente modo, calcola l'importo da pagare:
 - minimo 7,5 euro per i primi 80 scatti
 - più 0,6 euro/scatto per i successivi 60
 - più 0,5 euro/scatto per i successivi 60
 - più 0,4 euro/scatto per quelli oltre i 200