La automazione

Dalle macchine semplici alle macchine programmabili

Scheda **3**

Il calcolatore

1. Dalla calcolatrice al calcolatore

Alle calcolatrici non programmabili possiamo far eseguire solo una catena di operazioni di:

- addizione, moltiplicazione, … ,
- calcolo di altre funzioni il cui programma è stato incorporato dal costruttore,
- immagazzinamento o estrazione di dati da una "memoria".

A un ***calcolatore*** (o *computer* o *macchina calcolatrice programmabile*) possiamo far gestire l'esecuzione di un generico algoritmo che gli possiamo descrivere in un opportuno *linguaggio* "comprensibile" da esso: un algoritmo così descitto si chiama ***programma***.

Molti computer tascabili (o pocket computer) e tutti i calcolatori di maggiori dimensioni sono in grado di eseguire il seguente *programma (1.1)* scritto in linguaggio ***Basic***:

(1.1)

```
10 INPUT T      l'utente deve battere totale
20 K=100/T
30 INPUT D      l'utente deve battere dato
40 PRINT D*K
50 GOTO 30
```

È facile comprendere il significato di questo programma, che ha la forma di una sequenza di istruzioni numerate. PRINT in inglese significa "stampa"; GOTO deriva dall'inglese "go to", che significa "vai a". La parola "input", che abbiamo già usato, indica qualcosa che entra in una macchina (energia, informazioni, …); in questo caso INPUT viene usato per dire al calcolatore di attendere che l'utente batta un numero. Se non hai idea di come funzioni il programma puoi cliccare **qui**.

Per poter eseguire un programma come (1.1), il computer deve essere in grado di *memorizzare il testo del programma*, cioè, nel caso di (1.1), la seguente sequenza di caratteri, dove con "◊" e "↵" abbiamo indicato lo spazio bianco (pressione della barra spaziatrice) e l'"a capo", caratteri normalmente invisibili:

10◊INPUT◊T↵20◊K=100/T↵30◊INPUT◊D↵40◊PRINT◊D*K↵50◊GOTO◊30↵

Come le usuali CT memorizzano i numeri sotto forma di sequenze di bit, così un computer memorizza sotto forma di una sequenza di bit anche il testo del programma, utilizzando uno specifico codice.

Deve, inoltre, essere in grado di *leggere* (la sequenza di bit con cui ha codificato) *il programma e decidere, man mano, quale operazione eseguire*: una memorizzazione, un richiamo dalla memoria, un calcolo o una visualizzazione (operazioni che nelle CT non programmabili vengono comandate direttamente da tastiera), o un trasferimento dell'esecuzione a un altro punto del programma.

Per fare tutto ciò il computer deve avere:

- un "programma incorporato" per *codificare* il "programma battuto dall'utente",
- un dispositivo di memorizzazione in cui *registrare* il il programma codificato,
- un certo numero di *memorie-utente* da associare alle variabili (T, K, D, …) e
- un ulteriore "programma incorporato" per *tradurre* il programma codificato nell'azionamento dei vari dispositivi di calcolo, memorizzazione, … .

Il Basic risale al 1964. Esistono linguaggi di programmazione più sofisticati. Sotto a sinistra, si vede come il programma (1.1) può essere riscritto usando il ***QBasic***.. A destra è presentata una stesura del programma in un altro linguaggio, una versione di ***Pascal*** (il nome, *Pascal*, deriva da quello del famoso filosofo e scienziato francese - ➡ vedi - che, nella prima metà del 1600, inventò una delle prime calcolatrici da tavolo, per fare addizioni e sottrazioni).

(1.2)

```
INPUT Tot
k = 100/Tot

Introduzione:
    INPUT dato
    PRINT dato*k
GOTO Introduzione
```

```
program Percentuali;
var tot, k, dato: real;
label 10;
begin
   readln(tot);
   k:=100/tot;
10: readln(dato);
    write(dato*k);
    goto 10
end.
```

Esistono, infine, linguaggi di programmazione incorporati in tutti i browser. Con essi sono redatti i programmi che usiamo per cercare l'orario di un treno, prenotare un posto al cinema, …. Uno dei più usati tra essi è **JavaScript** (in breve, JS), di cui abbiamo già discusso **qui** e in cui sono redatti tutti gli "script" presenti in queste schede di lavoro. Il programma precedente assume la forma dello script **Perc**, visualizzato a lato dopo aver calcolato 62/720·100.

Metti *Totale* e *Dato*. Clicca e ottieni *Percentuale* (arrotonda il risultato!)

Totale = 720 **Dato =** 62

clicca **Percentuale =** 8.61111111111111 **(%)**

Sotto è illustrato come è stato realizzato il programma, che potete visualizzare usando un opportuno comando del browser ("visualizza sorgente pagina", "html", … o altro, a seconda del browser che state impiegando). Osservazione: i comandi e racchiudono il testo che si vuole mettere in *corsivo*.

```
<head>
<script language="javascript">
function Calcola() {
t = document.Perc.t.value; d = document.Perc.d.value;
document.Perc.p.value = Number(100/t*d)    }
</script>
</head>
```

(1.3)

```
<center>
<form name="Perc">
Metti <i>Totale</i> e <i>Dato</i>.   Clicca e ottieni <i>Percentuale</i><br>
(arrotonda il risultato!)<br>
Totale = <input type="text" name="t" value="" size=25>
Dato = <input type="text" name="d" value="" size=25><br>
<input type="button" value="clicca" onClick="Calcola()">
Percentuale =  <input type="text" name="p" value="" size=25> (%)<br>
</form>
</center>
```

Potete provare ad usare il programma (e ad esplorarne la struttura).

Qualche nota storica. I primi calcolatori programmabili (di enormi dimensioni) risalgono agli anni immediatamente successivi alla II guerra mondiale. Comunque l'idea di computer fu messa a fuoco da ***Alan Turing***, nel 1936, 15 anni prima delle realizzazione pratica dei primi modelli di computer: egli formalizzò il concetto generale di algoritmo e mise a punto il primo linguaggio di programmazione. I primi *personal computer* fanno la comparsa dopo il 1975, ma un computer di piccole dimensioni fu messo a punto dalla Olivetti già nel 1964. A fianco è raffigurato un modello di *HP85* (1979), il primo personal computer di piccole dimensioni (con incorporata una stampante, un lettore/registratore di dati e programmi su cassette magnetiche) ad avere una grossa diffusione in tutto il mondo. Qualche anno dopo (1984) si sviluppano (prima in ambito *Mac* e, da un certo punto in poi, soprattutto, in ambito *Windows*) personal computer che utilizzano il *mouse*.

Successivamente (in Italia a partire dal 1986) si diffonde l'uso di *Internet* e, poi, la possibilità di impiegare software via rete, in modo interattivo.
Dopo si sviluppano gli *schermi sensibili* al tatto, i cellulari diventano, per vari aspetti, dei piccoli personal computer, … Si sono diffusi negli ultimi anni anche i *tablet*, delle specie di piccoli personal computer senza "tastiera fisica" (hanno lo schermo sensibile al tatto). Sono comodi per leggere testi e scriverne (almeno per chi non ha problemi di dislessia), fare foto o filmati, ascolare brani musicali, …, ***ma*** non per fare cose più significative. Per dirla con Dan Gookin, "se nella vita digitale vi sentite solo di passaggio, allora potete cavarvela con uno smartphone o un tablet e non aver mai bisogno di un PC; se però avete bisogno di creare qualcosa, allora vi serve un computer" (da *PCs For Dummies*).

2. Ambienti di programmazione

Un *computer* è in grado di tradurre in operazioni-macchina un programma scritto in un qualunque linguaggio di programmazione, a patto che gli venga fornito in "input" anche un opportuno programma traduttore. La CPU di un computer può eseguire direttamente solo un programma che le arrivi sotto forma di una sequenza di bit che, a gruppi, rappresentino direttamente le operazioni-macchina da effettuare.

Il linguaggio (che ha come alfabeto i simboli 0 e 1) in cui sono scritti questi programmi è detto ***linguaggio macchina***.

I *linguaggi di programmazione* ***evoluti*** (come quelli usati sopra) non descrivono direttamente singole operazioni-macchina, ma, come abbiamo visto, • impartiscono comandi più sofisticati (a cui corrispondono più comandi del linguaggio-macchina), • usano come caratteri quasi tutti i simboli della tastiera e • indicano i comandi con nomi che ne richiamano il significato; per questo vengono detti ***linguaggi evoluti*** (o *di alto livello*). Per usarli il computer deve essere dotato di un ***ambiente di programmazione***, cioè un programma che contiene come sottoprogrammi:

- un ***programma redattore*** (***editor***) per registrare i caratteri battuti dall'utente; in genere è presente anche un ***help*** che richiama uso e significato dei comandi, propone esempi, … .

Un ***editor*** è una applicazione per leggere/elaborare testi (come NotePad/BloccoNote in Windows) che registra i documenti codificando i vari caratteri (lettere, cifre, virgole, spazi bianchi, e altri simboli) attraverso ***byte***, ossia sequenze di 8 bit (i caratteri codificabili in questo modo sono 256; infatti le diverse sequenze di 8 cifre che riesco a costruire con 0 e 1 sono 2·2·2·2·2·2·2·2 = 256). Il codice impiegato è chiamato **ASCII** (pronuncia: aski). Ad esempio "P" è codificato con 01010000, l'"a capo" con 00001101.

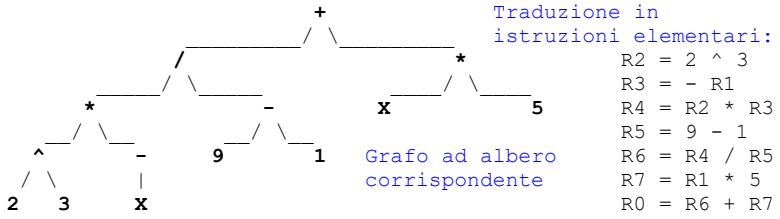
- un ***programma traduttore*** per tradurre il programma in linguaggio evoluto in una sequenza di bit che sia la versione in linguaggio macchina del programma.

Ecco come viene trasformata da un programma taduttore l'assegnazione (scritta in un linguaggio Basic) corrispondente alla formula a fianco:

$$w = \frac{2^{3}\cdot -x}{9 - 1} + x\cdot 5$$

```
ASSEGNAZIONE:          W = (2^3*-X)/(9-1)+X*5

variabili->REGISTRI:   R0 = (2^3*-R1)/(9-1)+R1*5
```



Grafo ad albero corrispondente

Traduzione in istruzioni elementari:

```
R2 = 2 ^ 3
R3 = - R1
R4 = R2 * R3
R5 = 9 - 1
R6 = R4 / R5
R7 = R1 * 5
R0 = R6 + R7
```

Nello stendere un programma in un linguaggio evoluto dobbiamo tener presenti delle regole di scrittura che garantiscano che il testo battuto sia effettivamente traducibile in linguaggio macchina. L'insieme di queste regole di scrittura viene detto ***sintassi***.

Anche nella lingua naturale esistono delle regole sintattiche. Ad esempio uno dei modi in cui si può comporre una *frase* in italiano è: *articolo+nome+verbo* dove *articolo*, *nome* e *verbo* devono essere raccordati in numero (sing./plu.) e, eventualmente, in genere (m/f), e rispettare altre eventuali condizioni (es.: davanti ai nomi maschili singolari se iniziano con *z*, *x*, *gn*, *pn*, *ps*, *s* seguita da consonante o *i* seguita da vocale si può mettere *uno* non *un*; negli altri casi si può mettere *un*, non *uno*). Ma si tratta di regole che spesso presentano eccezioni e su cui spesso esistono opinioni contrastanti (ad es.qualcuno sostiene che davanti a *pn* - pneumatico, pneumotorace, … - occorre, o si può, usare *un*). In realtà non si tratta di "regole" ma di *modelli* che usiamo per orientarci nella produzione/interpretazione dei messaggi verbali.

Poi, anche se ci esprimiamo in modo un po' sgrammaticato, in genere ci capiamo allo stesso (di fronte al cartello «attendi - lo cane morzica» non abbiamo difficoltà a interpretarlo come «Attenti: il cane morsica»).

Nel caso dei linguaggi di programmazione le regole sintattiche sono invece definite senza ambiguità ed eccezioni (per questo si parla di ***linguaggi formali***) e devono essere rispettate rigorosamente. L'*help* indica le regole da rispettare per costruire i programmi. Ad es. le seguenti istruzioni dei programmi del 1° paragrafo:

k = 100/Tot t = document.Perc.t.value

sono tutte ***assegnazioni***, che debbono avere la forma `variable = expression`. In altri linguaggi le assegnazioni debbono avere forme diverse: `variable <- expression` o `variable := expression` o …

Come nel caso del linguaggio comune la ***semantica*** si riferisce al significato di parole e frasi, così nel caso delle istruzioni essa si riferisce al loro significato, ossia a come esse vengono tradotte in linguaggio macchina. In particolare nel caso degli errori, mentre quelli sintattici impediscono la traduzione di un'istruzione in linguaggio macchina, quelli semantici avvengono in corso di esecuzione. Un esempio. Se definisco `f = function(x) x/(sqrt(4)-2)` e poi a questa istruzione faccio seguire `f(0)`, le istruzioni vengono eseguite e poi viene segnalato un errore "semantico" con un messaggio tipo `Not a Number`.

3. Automatizziamo qualche procedimento di calcolo in JavaScript

Vediamo ora qualche semplice programma realizzato in *JavaScript*. Incominciamo da quello considerato alla fine di §1, di cui abbiamo già riportato ➡ un esempio d'uso e il testo. Ha la struttura di una pagina Web: è un documento in HTML (**h**yper**t**ext **m**arkup **l**anguage: linguaggio per contrassegnare ipertesti). Abbiamo visto come esso appare visualizzato da un **browser** (applicazione per interpretare i documenti Html) e ne abbiamo visto il documento "**sorgente**", cioè il testo (redatto con un qualunque **editor**) comprendente il contenuto verbale del documento e i **tag** (contrassegni), ossia i comandi racchiusi tra parentesi angolari (< e >) che contengono indicazioni sui formati in cui visualizzare il contenuto verbale, le eventuali immagini da inserire, i collegamenti ad altri documenti, …, e indicano le azioni da eseguire, descritte tra i comandi "**script**". I tag "**head**" ("intestazione"), <head> e </head>, delimitano la parte del sorgente in cui sono contenute informazioni o comandi riferiti all'intero documento. Seguono comandi indicano ciò che viene visualizzato; i tag "**center**" fanno sì che esso sia raffigurato "centrato"; invece **
** (break) comanda un "a capo".

La parte del documento organizzata in caselle (*modulo*) è tra i tag "**form**"; forma un *oggetto* (ossia un *componente*) a cui, con la *proprietà* **name** è dato nome "Perc". Le caselle sono dei sotto-oggetti del modulo specificati con i tag "**input**"; a tre di esse, specificate come "**type**" *text*, viene dato un nome (t, d, p) ed è assegnata "**size**" (dimensione) 25 (sono delle celle che possono accogliere 25 caratteri); sull'altra casella, specificata come "**type**" *button*, torniamo tra poco; "**value**" assegna un eventuale valore iniziale agli oggetti precedenti; nel caso del "bottone" si tratta del nome che apparirà su di esso.

Nel caso del sotto-oggetto di tipo "button" l'attributo "**onClick**" specifica che se si verifica l'*evento* "pulsante cliccato" viene avviata l'azione (o *metodo*) "Calcola()" descritta dalla omonima **function** (così in Javascript vegono chiamati i *sottoprogrammi*). Il *programma* in senso stretto è racchiuso tra i comandi "**script**" (che in questo caso è descritto non nel corpo della pagina-web, ma nella intestazione) ed è costituito solamente dalla *function* Calcola().

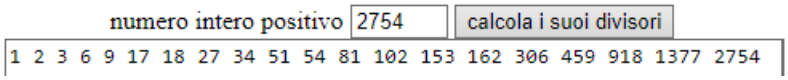
Il contenuto della function è delimitato da "**{**" e "**}**", e in questo è contenuto da una sola assegnazione. Essa legge i contenuti (ossia i valori: *value*) delle caselle di nome "d" e "t" del modulo di nome "Perc" e calcola il valore da mettere nella casella "p". Nelle assegnazioni è presente **document**: in questo modo viene indicata la pagina web attualmente visualizzata. In breve: *document.Perc.d.value* indica la proprietà "value" della proprietà (o sotto-oggetto) "d" della proprietà (o sotto-oggetto) "Perc" dell'oggetto "document".

Number serve per specificare che gli oggetti contenuti nelle caselle sono da intendere come "numeri", non come "testi": la somma di 3 e 4 viene intesa, altrimenti, come 34 invece che come 7.

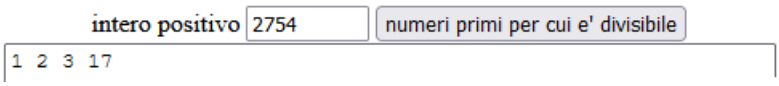
Ricordiamo, infine, che nelle istruzioni di JS lettere diverse solo per la dimensione (come "a" e "A") sono considerate diverse.

Abbiamo gà visto molti esempi di programmi in JS. Vediamone alcuni altri.

Lo script **divisori** consente di trovare tutti i numeri interi positivi per cui è divisibile un dato numero intero positivo (2754 nel caso esemplificato).



Lo script **divPrimi** ne trova invece tutti quelli che sono numeri **primi**, ossia che non sono divisibili per altri numeri positivi.



Lo script **scomponi** trova invece i numeri **primi** il cui prodotto è il numero indicato.

Tutti gli altri divisori si ottengono moltiplicando alcuni di questi numeri. Ad esempio posso ottenere 459 calcolando 3*3*3*17.

2 Trova i numeri primi per cui sono divisibili 12345678 e 123456789.

Provate ad eseguire lo script illustrato a lato, relativo al "*gioco del 3k+1*", a cui potete accedere dallo script **gioco**. Il gioco è spiegato nello script stesso. Esso termina quando si arriva ad 1.

Si arriva sempre ad 1 o ci sono numeri a partire dai quali il gioco non termina? Provate!

Quanti passi???

Questo è il gioco del "tre enne piu' uno".
Immetti un numero intero N:
- se esso è pari viene calcolato N/2, altrimenti 3*N+1;
- si ripete prendendo questo numero come nuovo N:
- si procede in modo analogo fino a quando non si arriva a 1.

Viene stampato il numero di passi per arrivare a 1.
Quanti sono?

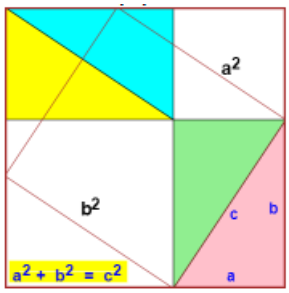
introduci un intero maggiore di uno e clicca [31] [clicca]

il numero di passi per arrivare a 1 è [106]

Prova ad es. con 31, 1000000000, 123456.
Non si è ancora dimostrato che in ogni caso il procedimento termina!!!

3 Vi sono input minori di 31 che danno un output maggiore di 106?

Gli script consentono di visualizzare anche immagini animate. Esamina lo script **Pitagora**.

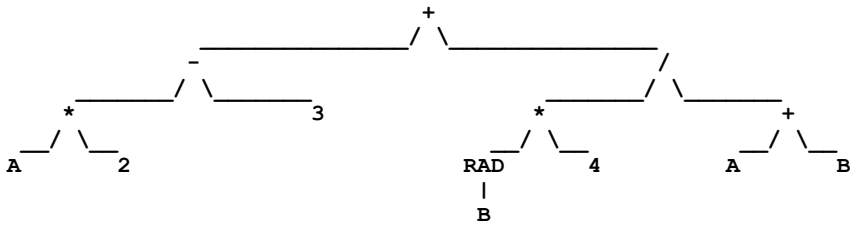


4. Esercizi

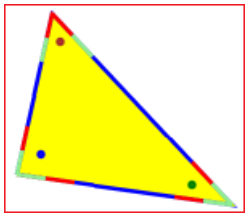
e1 Abbiamo descritto una funzione H nel modo indicato a lato. Cerca di descrivere H(x) sia con un grafo ad albero, sia nella scrittura ad un piano (quella che useresti in un programma)

$$H(x) = 3 + \frac{2 + 3x}{7} - \frac{5}{-2x}$$

e2 Scrivi il termine rappresentato dal grafo ad albero seguente sia in una scrittura "a più piani" che in una scrittura "ad 1 piano" (qui RAD indica la radice quadrata).



e3 Esamina lo script **SommaAngoli**. Che cosa viene spiegato e dimostrato con esso?



e4 Esamina lo script **Triangoli** e i relativi esempi. Poi impiegalo per determinare l'inclinazione di una strada che avanza orizzonatalmente di 19 metri e sale di 3 metri.

1) Segna con l'evidenziatore, nelle parti della scheda indicate, frasi e/o formule che descrivono il significato dei seguenti termini:
programma traduttore (§2), *sintassi* (§2), *tag* (§3), *form* (§3).
2) Su un foglio da "quadernone", nella prima facciata, esemplifica l'uso di ciascuno dei concetti sopra elencati mediante una frase in cui esso venga impiegato.
3) Nella seconda facciata riassumi in modo discorsivo (senza formule, come in una descrizione "al telefono") il contenuto della scheda (non fare un elenco di argomenti, ma cerca di far capire il "filo del discorso").

script: piccola CT grande CT isto isto con % boxplot striscia 100 60 ordina Grafici Perc divisori divPrimi scomponi gioco Pitagora SommaAngoli Triangoli