Prof. Paolo Molena / scuola media "Quintino di Vona" - Milano

Unità di Apprendimento ALGORITMI (11/2012)

area delle Informazioni / 1 media

1. Dal problema al programma:

.

1.a – introduzione

Algoritmo > ***codifica*** > programma con ling. ad alto livello > ***compilazione*** > programma in linguaggio assoluto La fase di avvio ***boot process***. Il **firmware** e le modifiche alla configurazione con la BIOS. Il **sistema operativo** MS-DOS e WINDOWS.

La programmazione

Il lavoro in diretta

1.b – problemi e procedimenti

L'analisi del problema e l'analisi dei dati. L'uso dei grafi di scomposizione

La procedura in linguaggio di progetto. L'uso di ***ottieni*** … ***facendo*** ….

Il programma in LIBERTY BASIC. L'uso di **LET** per le elaborazioni e **PRINT** per la comunicazione del risultato.

L'esecuzione e la verifica del risultato (l'ordine *Run*)

- il salvataggio su disco
- il caricamento da disco

1.c –problemi e classi di problemi

La generalizzazione del problema. L'uso dell'ordine **INPUT** nella richiesta di dati.

- **dati costanti**, i **dati parametrici**
- **problema specifico** e **problema generalizzato**
- la scheda di programma
- il percorso svolto
- *e ora al lavoro*

2. Procedimenti e dati:

2.a – conoscenza dei procedimenti, conoscenza dei dati nella mente dell'uomo

Rapporto tra procedimenti e dati

La padronanza dei procedimenti

2.b - conoscenza dei procedimenti, conoscenza dei dati nella RAM del calcolatore

L'area dei programmi e l'area delle variabili

Le variabili e il loro indirizzo

I nomi delle variabili

Esercitazioni sulle variabili

2.c – il lavoro umano

il lavoro dell'uomo: l'**operatore**, l'**analista** e il **programmatore**.

- in sintesi
- *e ora al lavoro*

3. Dati alfanumerici:

3.a – dal numero al carattere

Nei primi calcolatori solo numeri.

Il dato alfanumerico; la codifica dei caratteri in byte. I **codici ASCII** dei caratteri e l'uso di **ASC** e **CHR$**.

Dalla tastiera al monitor (la **memoria caratteri**, la **scheda grafica**).

3.b - dati alfanumerici nel programma

Le **situazioni comunicative** che si sviluppano nell'attività con il calcolatore. La chiara richiesta di dati e la chiara comunicazione. L'inserimento di ***commenti*** per l'operatore e l'uso di **CLS** e **LOCATE**. L' uso di ***annotazioni*** per il programmatore con **REM**.

- *e ora al lavoro*

3.c - le variabili alfanumeriche o "stringa"

L'inserimento di testi dentro una variabile. La struttura della variabili alfanumeriche. Lo **slicing** con **MID$**, **LEFT$** e **RIGHT$**. La "misurazione" delle variabili alfanumeriche con **LEN**. L'uso di **LOWER$**, **UPPER$** e **TRIM$**. Il passaggio da numerico ad alfanumerico con **VAL** e **STR$**.

- in sintesi
- *e ora al lavoro*

3.d - l'invio di dati verso le memorie secondarie

Salvare i dati presenti nell'area variabili della RAM utilizzando file sequenziali con codifica alfanumerica dei dati. L'operazione di **apertura** (OPEN), l'operazione di **scrittura** (**PRINT#**) o di **lettura** (**INPUT#** o **LINE INPUT#**), l'operazione di **chiusura** (**CLOSE**).

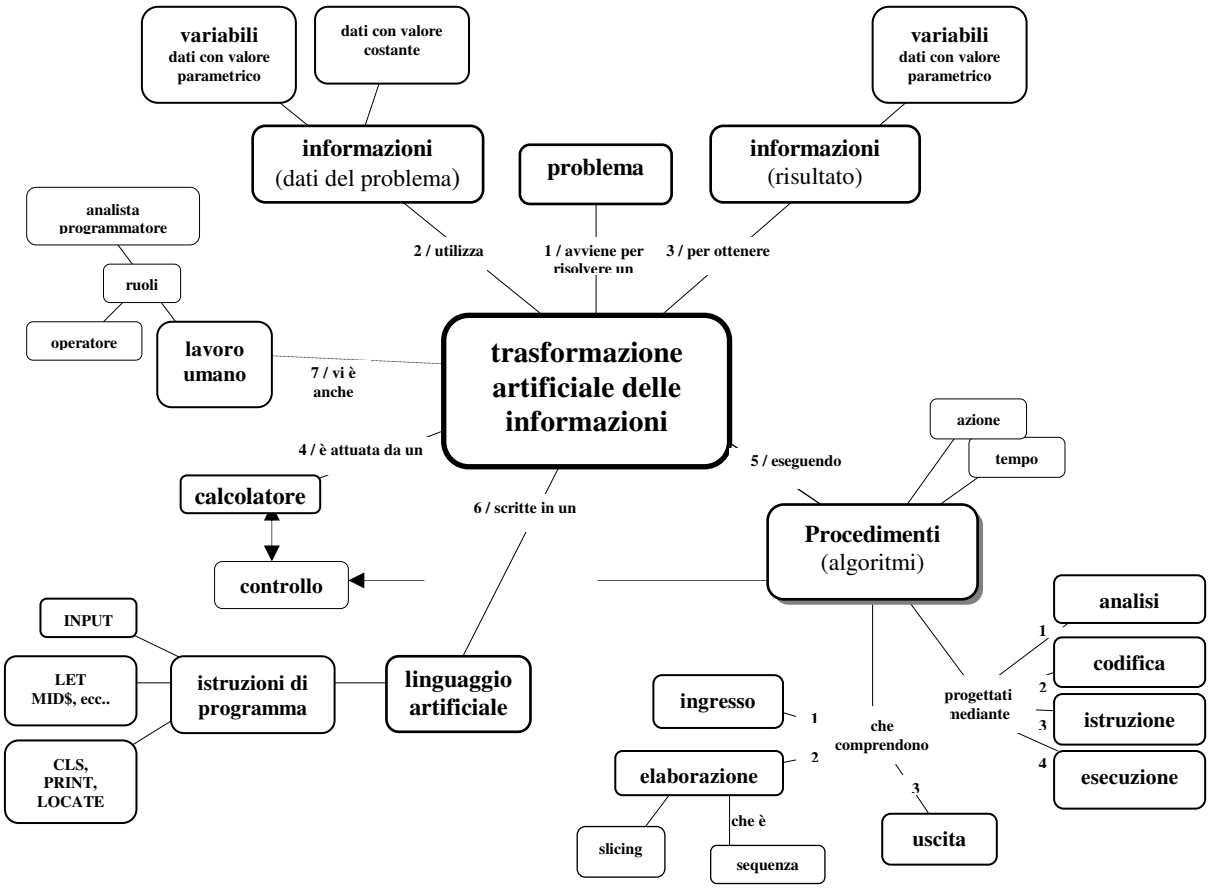
L'organizzazione dei file su disco: la struttura ad albero, la creazione e la rimozione di cartelle (**MKDIR**, **RMDIR**).

- *e ora al lavoro*

4. Risolvere i problemi con i fogli elettronici

L'impostazione del foglio di lavoro.
La risoluzione del problema
Verifica su un nuovo del problema
- *e ora al lavoro*

Questa mappa sintetizza i concetti che affronteremo nel corso di questa Unità di Apprendimento. Verbalizzane solo il primo livello



una trasformazione delle informazioni avviene …

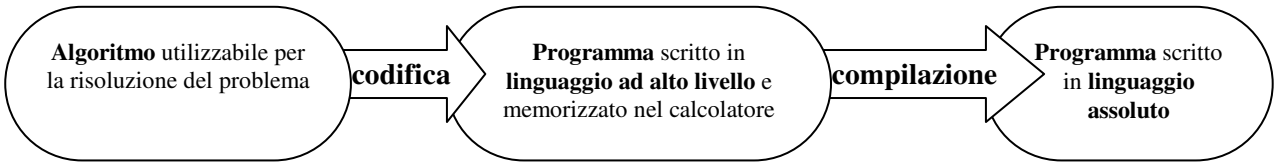
DAL PROBLEMA AL PROGRAMMA

Introduzione

In questa unità di apprendimento impareremo ad istruire un automa programmabile. Impareremo cioè, dato un problema, ad individuarne il procedimento di risoluzione (chiamato **algoritmo ***), a **codificarlo** in un linguaggio artificiale e a memorizzarlo nel calcolatore. Questo, dopo averlo **compilato** in **linguaggio assoluto**, potrà risolverlo ogni volta che sarà necessario.
In altre parole impareremo a **programmare un calcolatore** affrontando per ora semplici problemi, quelli che solitamente siamo abituati ad affrontare nell'attività di matematica.

* dal nome del matematico *Al Khwarizmi,* vissuto a Baghdad dal 750 al 830(?) ed autore del libro "Kitab al jabr wa'l mugabala" nel quale erano raccolte importanti nozioni di matematica di derivazione babilonese, indù ed alessandrina. Dalla distorsione del suo nome sono derivati i termini algoritmo e logaritmo e, dal titolo di una sua opera, il termine algebra.
Il termine **algoritmo** viene utilizzato per indicare quei procedimenti che, essendo espressi in modo rigoroso, dettagliato e non ambiguo, possono essere eseguiti anche da automi.

Il seguente grafico indica le operazioni da compiere per ottenere un programma da un algoritmo, programma grazie al quale sarà il calcolatore a risolvere il problema.

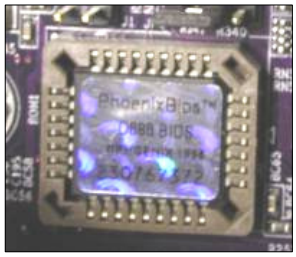


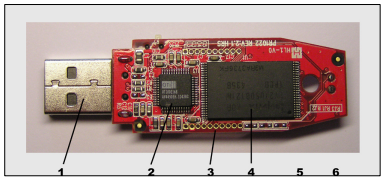
➢ l'ambiente di lavoro (MS-DOS e Windows)

All'accensione, i procedimenti presenti in **ROM *** (chiamati anche **firmware**) guidano il calcolatore nella fase di avvio, detta ***boot process***. Dopo i controlli preliminari sulla funzionalità dei componenti fondamentali (RAM, tastiera, dischi, porte, ecc…), il calcolatore, grazie al **BIOS** (Basic Input-Output System), verifica la collocazione e l'efficienza delle varie periferiche ed individua qual è il **disco di sistema** (cioè in quale memoria esterna si trova il sistema operativo da caricare per l'avvio). Nei computer IBM compatibili la ROM del BIOS contiene anche il POST, il primo programma che viene eseguito dopo l'accensione, coinvolto dunque nella fase di avvio (boot) del sistema di elaborazione.
Il BIOS è una memoria non volatile, in grado di memorizzare le modifiche alla configurazione.

*il termine ROM (Read Only Memory) viene ancor oggi utilizzato per indicare le memorie non volatili, che cioè mantengono i dati in memoria anche in mancanza di alimentazione. La memoria utilizzata è di tipo EEPROM. Lo stesso tipo di memoria riscrivibile che troviamo nelle **flash memory**. Questo tipo di memoria portatile è particolarmente indicato per la trasportabilità, proprio in virtù del fatto che non richiede alimentazione elettrica per mantenere i dati e che occupa poco spazio; è infatti molto usato nelle fotocamere digitali, nei lettori di musica portatili, nei cellulari, nei "chiavi" USB, ecc…

un chip che contiene il BIOS

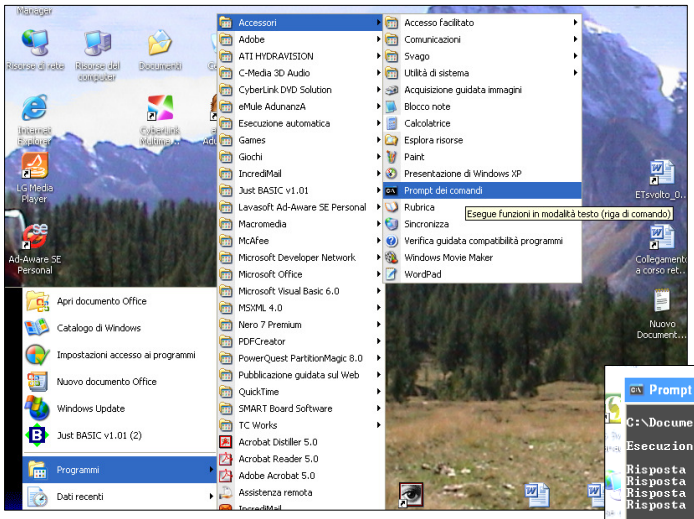




una "chiave USB

Individuato il disco di sistema (in genere è l'hard disk) il calcolatore carica in RAM il sistema operativo e poi lo avvia. Se si utilizzano i sistemi operativi della Microsoft, il calcolatore carica inizialmente le istruzioni del vecchio sistema operativo MS-DOS (MicroSoft Disk Operating System – diffuso negli anni '80). Poi si aggiungono le istruzioni di un sistema operativo più recente (Windows 98, … Windows XP). Sarà sempre possibile, per le operazioni di manutenzione della rete o la reinstallazione di sistemi operativi, tornare a lavorare in MS-DOS.

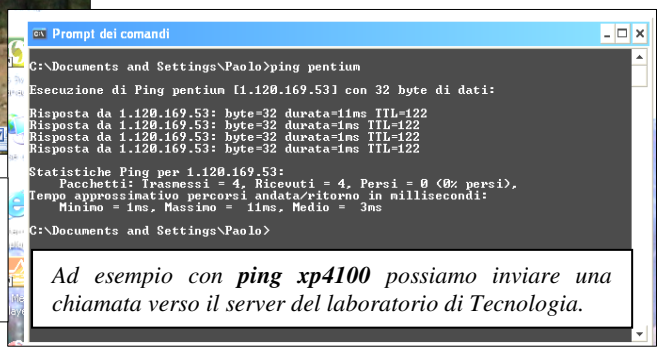
Per accedere al menù di modifica del BIOS è necessario premere all'avvio dell'elaboratore (dopo alcuni secondi che si è premuto il tasto di accensione) un tasto o un combinazione di tasti. Se l'elaboratore ha iniziato a caricare il sistema operativo non è più possibile (eccetto alcuni casi) variare le impostazioni del BIOS.
Questi i tasti più usati: Canc oppure Del /F2 /F10 Alt-F1
Una volta entrati nel menu di impostazione del BIOS è sufficiente seguire le indicazioni.



- con XP la finestra MS-DOS si chiama **prompt dei comandi** ed è presente in programmi>accessori
- con WIN 98 si chiama PROMPT MS-DOS e si accede direttamente da programmi

prove di collegamento in rete su finestra MS-DOS

per accertarsi che due calcolatori siano attivi in rete, dal menù programmi si apre la finestra di MS-DOS. E' una finestra scura in cui non vi sono icone e non si può usare il mouse (non era stato ancora inventato). Dunque ogni ordine va digitato sulla tastiera.
Scrivendo l'ordine PING seguito dal nome di un calcolatore, il nostro computer invia in rete quattro messaggi verso quel calcolatore. Se questo è attivo in rete avremo quattro risposte con i relativi tempi di arrivo.



*Ad esempio con **ping xp4100** possiamo inviare una chiamata verso il server del laboratorio di Tecnologia.*

In alto a sinistra abbiamo visto un **interfaccia utente*** di un moderno sistema operativo. Essa è basata su immagini che permettono, grazie al mouse, di selezionare le applicazioni desiderate. E' la **interfaccia grafica utente o GUI** (Graphical User Interface).
A destra il prompt dei comandi imita l'interfaccia utente del vecchio sistema operativo MS-DOS, detto a **linea di comando** perchè ogni comando doveva essere battuto sulla tastiera.

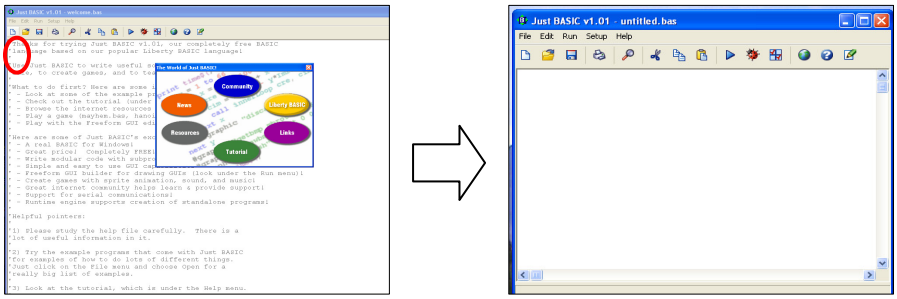
* viene definito **interfaccia utente** ciò che l'utente del calcolatore vede sul monitor.

Terminata l'installazione e l'avvio del sistema operativo il calcolatore è pronto a lavorare. Le istruzioni del sistema operativo servono infatti per preparare il calcolatore al lavoro.
Ora potremo avviare il software di un'applicazione (per videoscrittura, grafica, calcolo, giochi, ecc…). Potremo anche decidere di creare noi del nuovo software.

La programmazione

Mentre con i primi calcolatori questa attività veniva svolta utilizzando il linguaggio macchina (0 e 1), oggi è possibile utilizzare delle applicazioni che ci permettono di utilizzare un linguaggio vicino a quello naturale dell'uomo. Sarà poi lo stesso calcolatore a ***compilare*** il programma da noi realizzato, a tradurlo cioè in linguaggio macchina.
Noi utilizzeremo il LIBERTY BASIC. Si tratta di un applicazione realizzata negli Stati Uniti che eredita la semplicità dei primi linguaggi BASIC (come il GWBASIC utilizzabile in ambiente MS-DOS) pur essendo studiato per realizzare applicazioni in ambiente WINDOWS. Per ora ci permette di evitare il più complesso VISUAL BASIC.

Se nel nostro calcolatore è stato installato LIBERTY BASIC potremo avviarlo dal menù programmi o con un doppio click sull'icona di collegamento.



La finestra di benvenuto (welcome.bas) con all'interno una finestra di collegamento con il sito di LIBERTY BASIC su cui potremo trovare vari servizi di consulenza (tutti in inglese). Dopo aver chiuso la finestra del sito, un doppio click su ***new file*** chiude la finestra di benvenuto e apre la **finestra di programmazione** * chiamata ***untitled.bas***. Questo il nome provvisorio che LB ha dato al nostro programma. Potremo sostituirlo al primo salvataggio.

* La finestra di programmazione viene utilizzata per scrivere il programma che dovrà poi essere eseguito. E' simile ad una pagina di videoscrittura. Al momento dell'esecuzione le istruzioni verranno poi eseguite nell'ordine in cui sono state scritte.

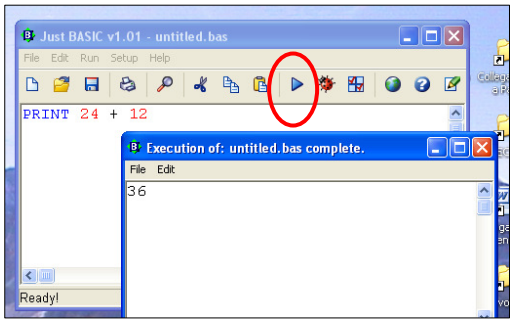
Il lavoro in diretta

Anche se abbiamo a disposizione un calcolatore, potremo sempre utilizzarlo come una calcolatrice. Possiamo usarlo cioè per fargli eseguire un singolo ordine invece che un intero procedimento. Questo modo di lavorare viene chiamato **lavorare in diretta**.
Se, facendo precedere il calcolo da effettuare dall'ordine PRINT (scrivi), batterò sulla tastiera:

PRINT 24 + 12

seguito da un click sul bottone di esecuzione ▷ *(Run)*

comparirà la **finestra di esecuzione** * con il risultato dell'operazione.



* La **finestra di esecuzione** si apre quando il calcolatore sta eseguendo un programma ed è lo strumento di comunicazione tra il calcolatore e l'operatore (la persona che utilizzerà l'applicazione da noi realizzata). Essa è modificabile nelle dimensioni e nell'aspetto e al suo interno potremo collocare altri oggetti grafici utili nella comunicazione con l'operatore. Infatti il sistema operativo WINDOWS (finestre) è chiamato così proprio per la possibilità di utilizzo di questi oggetti grafici. Noi però ci occuperemo di questi aspetti in terza con l'U.di A. SISTEMI.

Facendole precedere sempre dall'ordine PRINT, sarà possibile far risolvere al calcolatore intere espressioni matematiche, ricordando però che, in BASIC, i segni per gli operatori algebrici a volte non corrispondono a quelli che noi utilizziamo normalmente in matematica.

Dovremo pertanto memorizzare i segni collocati nella tabella qui di fianco ricordando che per indicare i numeri decimali al posto della virgola si dovrà usare il punto e che è possibile usare solo le parentesi tonde. E' però possibile usarle in modo gerarchico, collocando cioè all'interno le parentesi con i calcoli che vanno eseguiti prima.

Il BASIC: segni per gli operatori algebrici
(in istruzioni di assegnazione)

=	assegna un valore ad una variabile
-	segno dei numeri negativi
^	elevato a potenza
*	moltiplicato
/	diviso
+	più
-	meno

Se ad esempio batterò sulla tastiera:

PRINT 8*(4+2*(12+2/4)) seguito da ▷

sulla finestra di esecuzione comparirà il numero **232**
Dovremo infine ricordare che le frazioni vanno battute sullo schermo sotto forma di divisione.

Ad esempio: $\frac{4}{5}$ si scriverà **4/5**

Tuttavia il calcolatore non è stato costruito per funzionare come calcolatrice, ma per **essere programmato**, cioè per essere istruito dall'uomo per risolvere problemi.

Problemi e procedimenti

l'analisi del problema e l'analisi dei dati (l'uso dei grafi di scomposizione)

Ma come fa il calcolatore a risolvere problemi? Prendiamo dunque un problema tratto da un libro di matematica e vediamo cosa fare affinché esso possa essere gestito e risolto da un calcolatore

> *Un fruttivendolo acquista 100 kg. di aranci a 1 euro al chilo. Li rivende a 2 euro al chilo, avendone però scartati 10 kg. perché avariati. Quanto guadagna il fruttivendolo?*

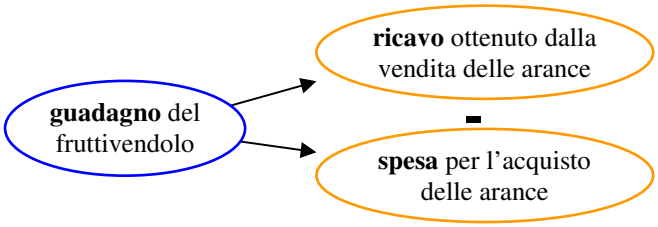
Iniziamo ad evidenziare i dati inseriti all'interno del testo.

Un fruttivendolo acquista **100 kg.** di aranci a **1 euro** al chilo. Li rivende a **2 euro** al chilo, avendone però scartati **10 kg.** perché avariati. Quanto **guadagna il fruttivendolo**?

il colore rosso indica _____

il colore viola indica _____

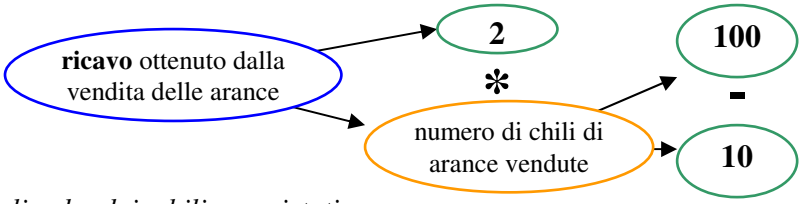
Procediamo ora nell'**analisi del problema**, un attività che, partendo dalla riflessione su ciò che dobbiamo ottenere (il guadagno), ci permette di ottenere il procedimento necessario per calcolarlo.
Per realizzarla utilizzeremo i **grafi di scomposizione**. I nomi dei dati vengono scritti all'interno di un ovale e delle frecce ci portano, partendo dal dato finale, ai dati iniziali forniti dal testo del problema.
Dunque si parte dal risultato e si procede ad una continua scomposizione sino ad arrivare ai dati iniziali del problema.



Procediamo dunque ricordando che, in questo tipo di problemi, il ***guadagno è ottenuto facendo ricavo meno la spesa*** e adattiamo questi dati al contesto del problema.
Passiamo poi a scomporre il ricavo ottenuto dalla vendita degli aranci.

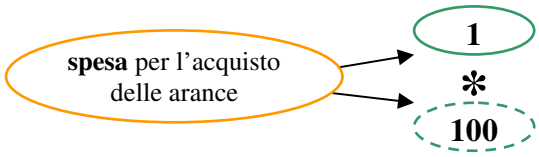
Noi conosciamo il prezzo di vendita al chilo (2 euro) che va moltiplicato per il numero di chili venduti.

Questi possono essere ottenuti togliendo dai chili acquistati (100) i chili scartati (10).



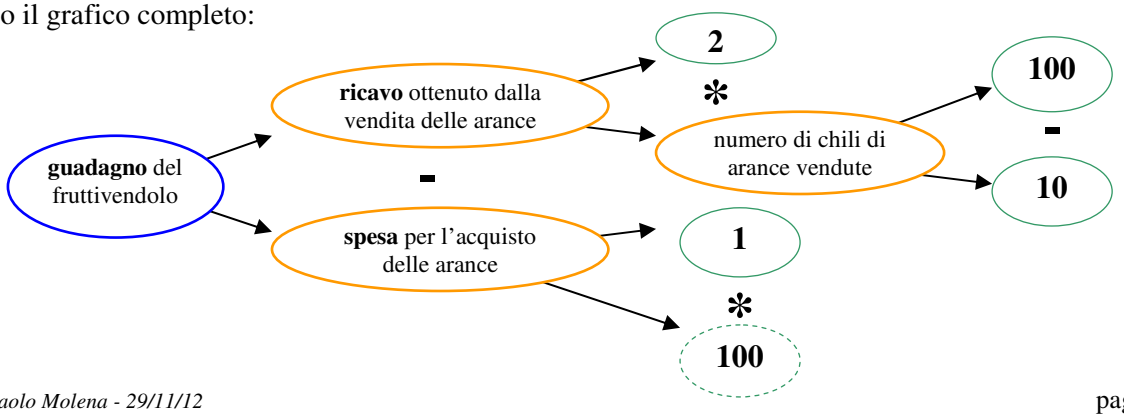
Viene dunque completato questo ramo del grafo arrivando ai dati iniziali.

Si completa poi il grafo scomponendo anche la spesa ed arrivando così ad ottenere il grafo completo.

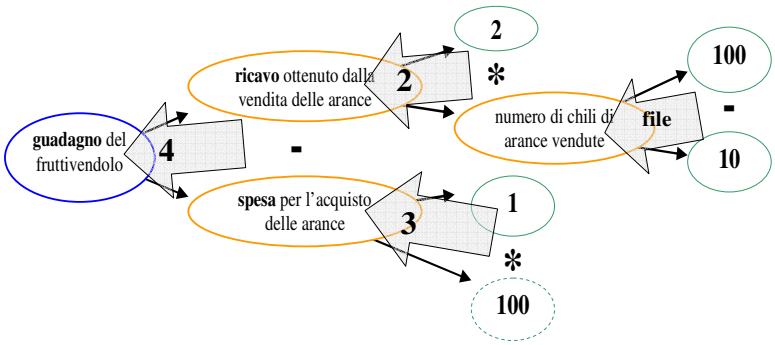


il tratteggio indica che il dato è già presente

Ecco il grafico completo:



la procedura in linguaggio di progetto

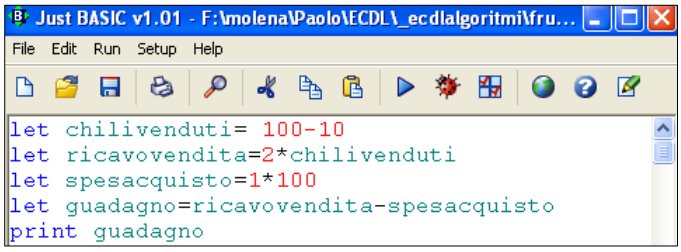


Osservando il grafico così ottenuto potremo osservare che da esso si può ricavare facilmente la procedura da utilizzare per risolvere il problema. Questa volta si parte da destra, cioè dai dati forniti dal problema, per arrivare ad ottenere il risultato.
Per scrivere il procedimento utilizzeremo il **linguaggio di progetto**. Non è un vero e proprio linguaggio; si tratta di scrivere il procedimento in italiano seguendo però regole di sintassi che richiamano quelle che utilizzeremo poi in BASIC. Le istruzioni di calcolo vanno scritte nel seguente modo:

ottieni *nome del dato che si vuole ottenere* **facendo** *elaborazione da compiere*
Scrivi ora le istruzioni che compongono il procedimento:
ottieni numero di chili di arance vendute facendo 100 - 10
ottieni _____
ottieni _____
ottieni _____
visto che stiamo scrivendo un procedimento che dovrà poi essere risolto da un calcolatore, è necessario che questo alla fine comunichi il risultato (**uscita dei dati**). Dunque il procedimento terminerà con:
scrivi _____

il programma in LIBERTY BASIC

Apriamo ora la finestra di programmazione. Dovremo tradurre in programma la procedura appena realizzata in modo che possa essere gestita da un calcolatore.
Per ora ci basta sapere che in BASIC l'ordine **ottieni** si traduce in **let,**
facendo viene invece tradotto con il segno **=**
Infine **scrivi** viene tradotto con **print**

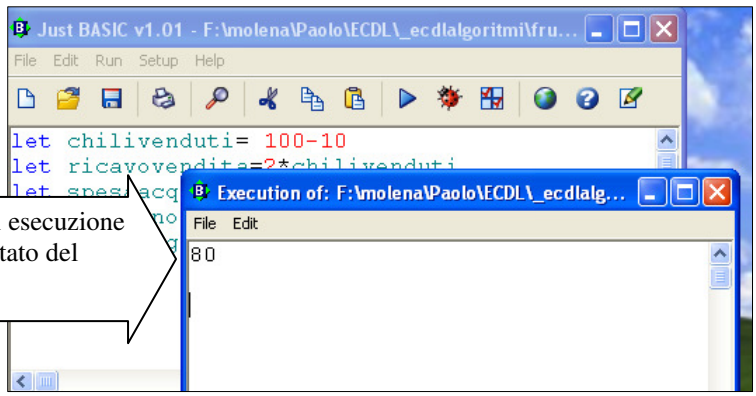


In attesa di approfondire le modalità con cui il calcolatore memorizza i dati (lo faremo nella prossima fase), ci limitiamo a dire che nel programma si chiama **variabile** lo spazio di memoria che ospita un dato.
I nomi delle variabili non devono contenere spazi ed è bene non siano troppo lunghi.
Ad esempio potremo sostituire:

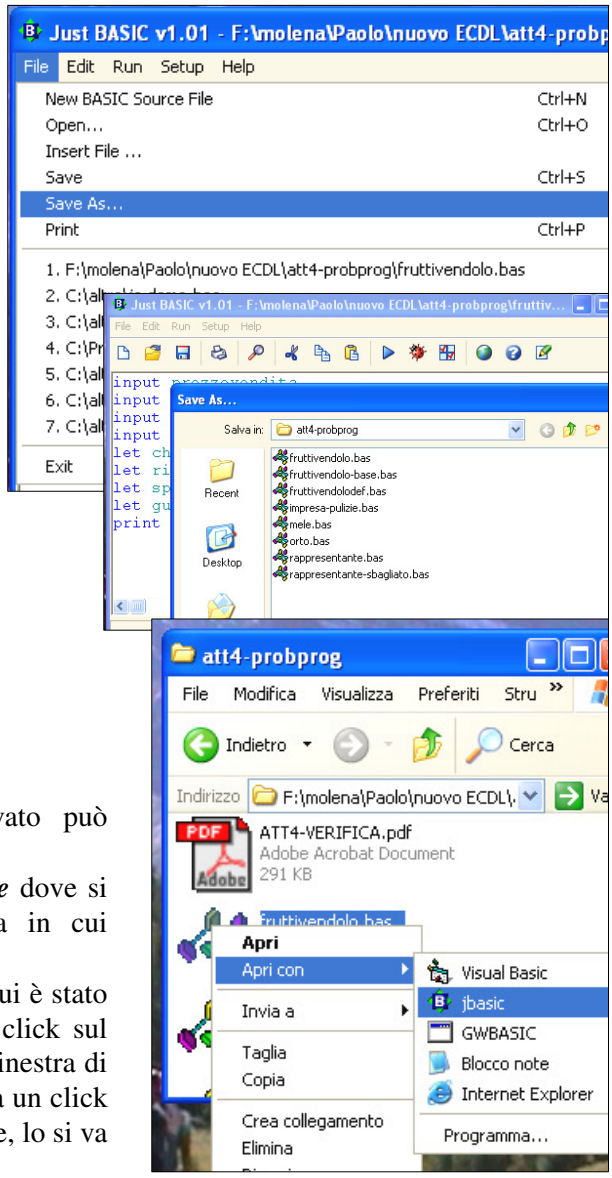
ricavo dalla vendita degli aranci con **ricavovendita**.

l'esecuzione del programma

Completata la scrittura del ***programma*** l'ordine ▷ *(Run)* permette la sua esecuzione.
Compare la finestra di esecuzione sulla quale vi è il numero **80** che rappresenta il dato finale (guadagno del fruttivendolo). Ci sarà facile verificare se il risultato è corretto.



- il salvataggio su disco



Se si vuole conservare il programma realizzato bisognerà salvarlo su disco, visto che tutte le informazioni presenti nella memoria RAM vengono cancellate allo spegnimento del calcolatore.
Dal menù ***file*** si potrà selezionare ***Save As…*** e appare la finestra di salvataggio dove, con le solite modalità, potremo dare un nome al programma e selezionare la cartella su cui salvarlo (il calcolatore propone sempre la cartella che in quel momento è operativa..
E' bene dare ai programmi nomi che ne ricordino il contenuto.
E' bene anche ricordare che *LB* salva i programmi in formato testo aggiungendo sempre, al nome che noi abbiamo scelto, il suffisso ***.bas*** (questo anche se noi dimentichiamo di aggiungerlo).
I programmi *LB* sono leggibili utilizzando Blocco Note. Potremo anche aprire un testo scritto in Blocco Note utilizzando *LB*

- il caricamento da disco

L'apertura di un programma precedentemente salvato può avvenire in due modi:

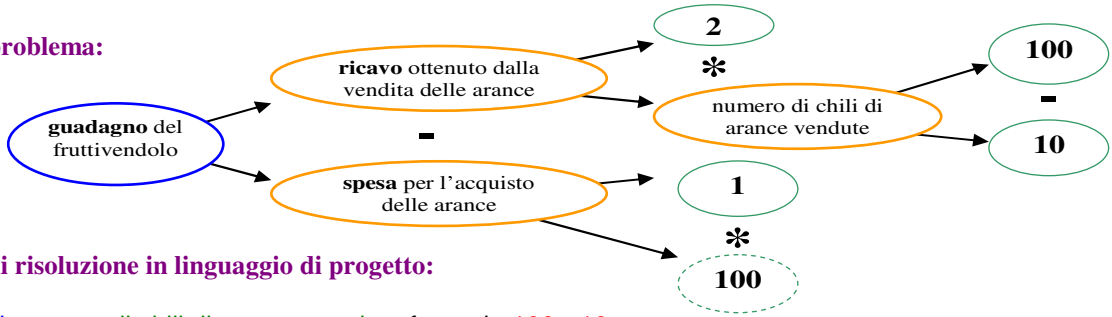
- dopo avere aperto *Liberty Basic* si va sul menù ***file*** dove si seleziona ***Open …*** e appare la finestra di ricerca in cui selezionare cartella e file da aprire
- si seleziona in ***Risorse del computer*** la cartella in cui è stato salvato il programma e, dopo averlo evidenziato (un click sul tasto sinistro) con un click sul tasto destro si apre una finestra di scelta. Si va su ***Apri con …*** e, sulla nuova finestra, si fa un click su *Liberty Basic.* Se su questa finestra LB non è presente, lo si va a cercare selezionando ***Programma …***

Rivediamo ora in sintesi il percorso svolto

Problema da risolvere:

Un fruttivendolo acquista 100 kg. di aranci a 1 euro al chilo. Li rivende a 2 euro al chilo, avendone però scartati 10 kg. perché avariati. Quanto guadagna il fruttivendolo?

Analisi del problema:



Procedura di risoluzione in linguaggio di progetto:

ottieni numero di chili di arance vendute facendo 100 - 10
ottieni ricavo ottenuto dalla vendita delle arance facendo numero di chili di arance vendute * 2
ottieni spesa per l'acquisto delle arance facendo 1 * 100
ottieni guadagno del fruttivendolo facendo ricavo ottenuto dalla vendita delle arance - spesa per l'acquisto delle arance
scrivi guadagno del fruttivendolo

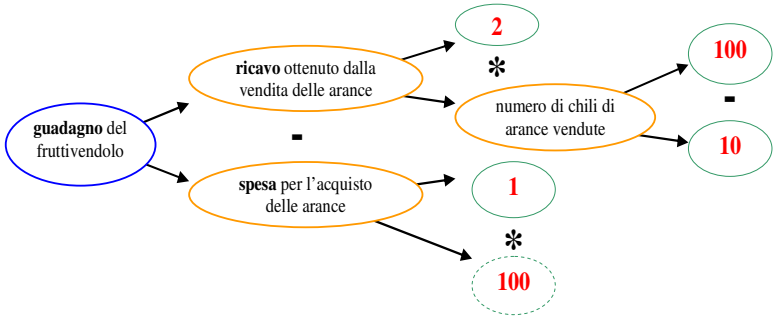
Dalla procedura al programma:

ottieni numero di chili di arance vendute facendo 100 - 10	let chilivenduti = 100 - 10
ottieni ricavo ottenuto dalla vendita delle arance facendo numero di chili di arance vendute * 2	let ricavovendita = chilivenduti * 2
ottieni spesa per l'acquisto delle arance facendo 1 * 100	let spesacquisto = 1 * 100
ottieni guadagno del fruttivendolo facendo ricavo ottenuto dalla vendita delle arance - spesa per l'acquisto delle arance	let guadagno = ricavovendita - spesacquisto
scrivi guadagno del fruttivendolo	print guadagno

Problemi e classi di problemi
la generalizzazione del problema

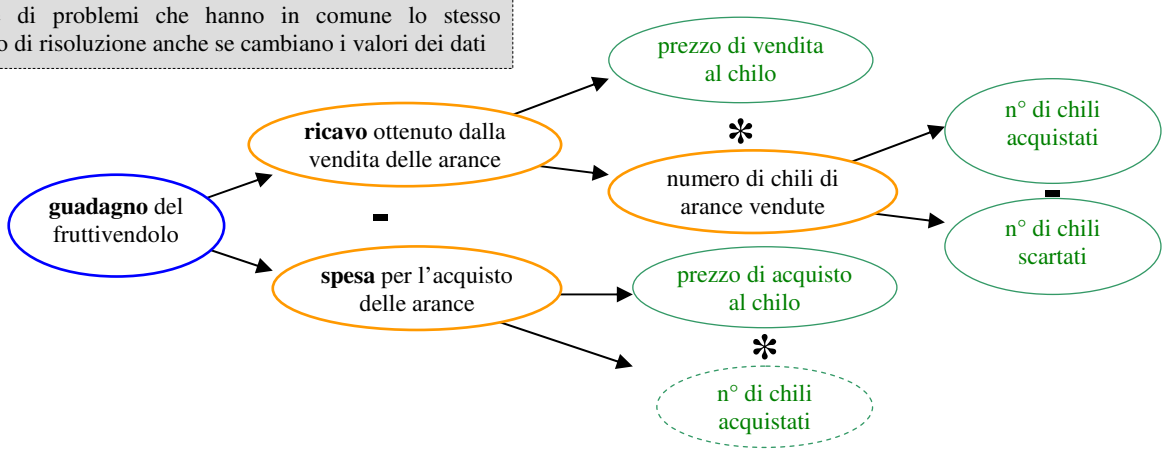
A questo punto possiamo chiederci se valeva la pena utilizzare il calcolatore per risolvere questo problema o sarebbe stato molto più veloce risolverlo in modo tradizionale. La risposta è evidente. Ma allora quando è conveniente utilizzare il calcolatore?
Se noi riuscissimo ad utilizzare il programma appena scritto non una sola volta, ma tutte le volte che il fruttivendolo acquista della frutta e la rivende, il tempo impiegato a istruire il calcolatore sarebbe stato ben speso.

Riguardando il grafo di scomposizione risulta evidente che la limitazione di utilizzo è data dal fatto che i dati iniziali hanno un valore assegnato e predefinito.



Sostituendo il valore del dato con il suo nome e potendo cambiare valore ad ogni esecuzione il procedimento sarà utilizzabile ogni volta che è necessario risolvere quel tipo o ***classe di problemi*** * .

* una serie di problemi che hanno in comune lo stesso procedimento di risoluzione anche se cambiano i valori dei dati



Ma il calcolatore come fa a sapere quali sono i valori dei dati?
Li dovrà chiedere prima di ogni esecuzione.
L'istruzione **input** (**leggi** in linguaggio di progetto) permette al calcolatore, nel momento dell'esecuzione di un programma, di richiedere il valore dei dati.
Qui di seguito troviamo, completata con l'**ingresso dei dati**, la procedura in linguaggio di progetto:

leggi prezzo di vendita al chilo
leggi n° di chili acquistati
leggi n° di chili scartati
leggi prezzo di acquisto al chilo
ottieni numero di chili di arance vendute facendo n° di chili acquistati - n° di chili scartati
ottieni il ricavo ottenuto dalla vendita delle arance facendo prezzo di vendita al chilo * numero di chili di arance vendute
ottieni la spesa per l'acquisto delle arance facendo prezzo di acquisto al chilo * n° di chili acquistati
ottieni il guadagno del fruttivendolo facendo il ricavo ottenuto dalla vendita delle arance - la spesa per l'acquisto delle arance
scrivi il guadagno del fruttivendolo

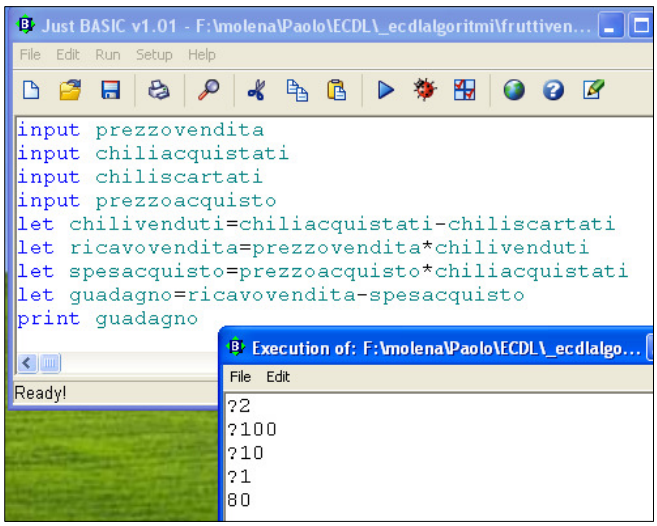
che trasformata in programma sarà:

anche in questo caso i nomi dei dati utilizzati nel linguaggio di progetto verranno sostituiti con nomi di variabili più corti e privi di spazi

ingresso dei dati	input prezzovendita input chiliacquistati input chiliscartati input prezzoacquisto
elaborazione	let chilivenduti = chiliacquistati - chiliscartati let ricavovendita = prezzovendita * chilivenduti let spesacquisto = prezzoacquisto * chiliacquistati let guadagno = ricavovendita - spesacquisto
uscita	print guadagno

l'esecuzione

E' venuto il momento di provare il nostro nuovo programma. Per la prima volta lo proviamo con i valori che abbiamo già utilizzato in modo da poter facilmente controllare il corretto funzionamento.
Con ▷ *(Run)* avviamo il programma.
Il calcolatore, facendo comparire un punto interrogativo sulla finestra di esecuzione, ci chiede i valori dei singoli dati nell'ordine che abbiamo previsto nel programma. Ad ogni richiesta noi scriviamo il valore seguito da Invio .
Alla fine viene scritto il risultato che ci conferma la correttezza del programma.
Successivamente con valori differenti, il calcolatore darà nuovi risultati.



E se non funziona?
Un'attività tipica della programmazione è la ricerca dell'errore. Dunque se non compare il risultato atteso evitare il solito *"prof. non funziona ...."* ma cercare per proprio conto quello che non va. E' bene tenere presente che il 90% degli errori deriva da un errore di scrittura dei nomi delle variabili. Per il calcolatore ricavovendita non ha niente a vedere con ricavovendite o con Ricavovendite. Se in ricavovendita c'è il risultato di un calcolo in ricavovendite c'è 0.

dati costanti, dati parametrici

Osserviamo ora il seguente problema:

> *Un fruttivendolo acquista un certo numero di chili di arance ad un dato prezzo al chilo. Li rivende ad un prezzo al chilo maggiore dopo aver scartato un certo numero di chili di arance avariate. Quanto guadagna il fruttivendolo?*

E' il problema che abbiamo risolto con il programma appena realizzato. In essa i valori dei dati sono stati sostituiti dai loro nomi.

La differenza che abbiamo individuato tra le due versioni dello stesso tipo di problema è che nel primo caso il valore dei dati era già definito. In questo caso si parla di **dati con valore costante**.

nome del dato	*tipo*	*valore*
n° di chili acquistati	*costante*	*100*
n° di chili scartati	*,,*	*10*
prezzo di acquisto al chilo in euro	*,,*	*1*
prezzo di vendita al chilo in euro	*,,*	*2*

Nel secondo caso il valore dei dati non è stabilito; spetta a chi eseguirà il procedimento (e dunque al calcolatore) chiederne il valore all'inizio dell'esecuzione. In questo caso si parla di ***dati con valore parametrico***. I dati che il calcolatore richiede nella fase d'ingresso si chiamano **parametrici d'entrata**, quelli che comunica nella fase di uscita si chiamano **parametrici d'uscita**, gli altri si chiamano **parametrici interni**.

nome del dato	*tipo*	*valore*
n° di chili acquistati	*par. d'entrata*	*?*
n° di chili scartati	*,,*	*?*
prezzo di acquisto al chilo in euro	*,,*	*?*
prezzo di vendita al chilo in euro	*,,*	*?*

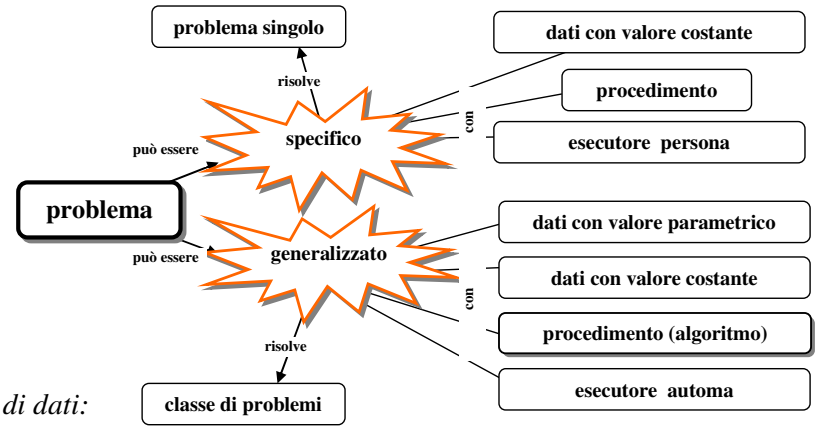
… in sintesi

Ragionando su quanto abbiamo appena visto proviamo ora a dare alcune definizioni:

il **problema specifico** è il classico problema che siamo stati abituati a risolvere sin dalle elementari, che ci vede come esecutori e dove l'importante è, oltre ad individuare il procedimento di risoluzione, individuare il risultato. Caratteristica di questo tipo di problema è la presenza di dati con valore costante e dunque il fatto che vi sarà una sola esecuzione

il **problema generalizzato** ha come esecutore il calcolatore (spetta a lui trovare il risultato). Il nostro compito è individuare la procedura di risoluzione e comunicarla al calcolatore. Caratteristica di questo tipo di problema è la prevalenza dei dati con valore parametrico; sarà il calcolatore a chiederne il valore che potrà essere diverso ad ogni esecuzione. Esso dunque serve a risolvere non un problema singolo, ma una classe di problemi

Esamina dunque lo schema riassuntivo collocato a fianco:



Esercizio: distinguere nomi e valori di dati:

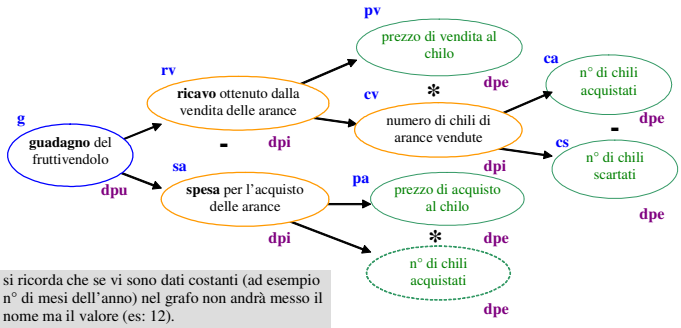
nome del dato	*valore del dato*	*nome del dato*	*valore del dato*
colore dei capelli	*rosso*		

... la scheda di programma

Le schede di programma ci permetteranno, d'ora in poi, di descrivere tutte le fasi di lavoro che partono dalla lettura del problema specifico per arrivare al programma per il calcolatore.
Si parte dal testo del ***problema specifico*** e si fanno i seguenti passaggi:

- scrittura del testo del ***problema generalizzato***
- realizzazione del ***grafo di scomposizione***
- ***analisi dei dati*** (sul grafo)
- scrittura della ***procedura in linguaggio di progetto***
- individuazione, sul grafo, dei ***nomi delle variabili*** (facoltativo)
- scrittura del ***programma***

Problema specifico 0	Problema generalizzato 0
Un fruttivendolo acquista 100 kg. di aranci a 1 euro al chilo. Li rivende a 2 euro al chilo, avendone però scartati 10 kg. perché avariati. Quanto guadagna il fruttivendolo.	Un fruttivendolo acquista un certo numero di chili di arance ad un dato prezzo al chilo. Li rivende ad un prezzo al chilo maggiore dopo aver scartato un certo numero di chili di arance avariate. Quanto guadagna il fruttivendolo?



si ricorda che se vi sono dati costanti (ad esempio n° di mesi dell'anno) nel grafo non andrà messo il nome ma il valore (es: 12).
Nell'analisi dei dati andrà indicato con **dc**

procedura in linguaggio di progetto

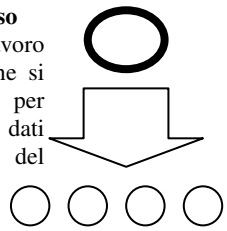
```
leggi prezzo di vendita al chilo
leggi n° di chili acquistati
leggi n° di chili scartati
leggi prezzo di acquisto al chilo
ottieni numero di chili di arance vendute facendo
      n° di chili acquistati - n° di chili scartati
ottieni il ricavo ottenuto dalla vendita delle
      arance facendo prezzo di vendita al chilo *
      numero di chili di arance vendute
ottieni la spesa per l'acquisto delle arance
      facendo prezzo di acquisto al chilo * n° di
      chili acquistati
ottieni il guadagno del fruttivendolo facendo il
      ricavo ottenuto dalla vendita delle arance -
      la spesa per l'acquisto delle arance
scrivi il guadagno del fruttivendolo
```

programma

```
input pv
input ca
input cs
input pa
let cv = ca - cs
let rv = pv * cv
let sa = pa * ca
let g = rv - sa
print g
```

… il percorso svolto sinora

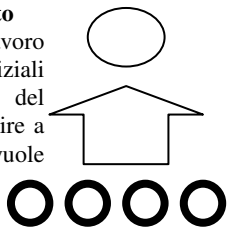
dall'alto al basso
in questa fase di lavoro si parte da ciò che si vuole ottenere per scendere sino ai dati presenti nel testo del problema



l'analisi del problema

si approfondisce la conoscenza della situazione problematica; si legge con cura il testo del ***problema generalizzato*** comprendendo che cosa si vuole ottenere e che cosa si ha a disposizione per poterlo ottenere
- si compila il grafo di scomposizione
- si fa l'analisi dei dati

dal basso in alto
in questa fase di lavoro si parte dai dati iniziali presenti nel testo del problema per risalire a ciò che si vuole ottenere



l'individuazione del percorso

si individua la sequenza di azioni necessaria per ottenere il risultato desiderato partendo dai dati iniziali
- si realizza la procedura in linguaggio di progetto

la codifica

le singole azioni vanno tradotte in un linguaggio artificiale comprensibile dall'automa
- si scrive il programma sulla scheda

l'istruzione del calcolatore

le singole istruzioni, scritte nel linguaggio artificiale, vengono scritte sulla tastiera e memorizzate nel calcolatore ***(attività di istruzione)***

l'esecuzione

l'automa eseguirà tutta la sequenza di azioni nel momento in cui vi sarà dato l'**avvio** al programma ***(attività di esecuzione)***

e ora al lavoro su nuovi problemi

Ecco due nuovi testi di problemi specifici. Ricopiali sulle schede di programma date dall'insegnante e svolgi autonomamente tutto il percorso. Sostituisci subito, ai valori dei dati d'ingresso, i loro nomi e compila così il riquadro che ospita il testo del problema generalizzato. Seguono poi il grafo di scomposizione, l'analisi dei dati, la procedura in linguaggio di progetto e il programma.
L'ultimo passo è quello di scrivere il programma sul calcolatore, eseguirlo utilizzando i dati del problema specifico e verificare che il dato di uscita sia corretto.
Attenzione! Non sempre vale l'impostazione ***guadagno = ricavo – spesa*** *che abbiamo usato nel problema precedente.*

Problema specifico 5

Un'agenzia immobiliare ha comperato un appartamento di 80 mq per 20.000 €. Lo ristruttura spendendo in tutto 8000 €.
Se poi lo ha rivenduto a 400 € al metro quadrato, quanto ha guadagnato?

Problema specifico 6

Un'impresa di pulizie paga un suo dipendente in base al numero di ore di lavoro effettuate. Sapendo che viene pagato 10 euro l'ora e che nel corso del mese ha lavorato 200 ore calcolare la paga mensile del dipendente.

Problema specifico 1

Il rappresentante di una ditta ha già percorso ***1200 km*** in auto. La città di destinazione dista ***2000 km*** dalla città da cui è partito. Quanti giorni impiegherà ancora se percorre abitualmente ***100 km*** al giorno?

Problema specifico 2

Una cassetta di mele pesa piena ***19.5 kg*** e vuota ***1.5 kg***. Se una mela pesa circa ***0.2 kg***, quante mele ci sono nelle cassette?

Problema specifico 3

Un negoziante ha acquistato all'ingrosso, per poi venderli al dettaglio, ***36 metri*** di tela. Il ricavo totale dalla vendita al dettaglio è stato di ***180 euro***. Sapendo che il suo guadagno netto è stato di ***45 euro***, calcolare quale è stato per il negoziante il costo di ogni metro lineare di tela all'ingrosso.

Problema specifico 4

Un contadino deve recintare un orto che ha forma rettangolare con le dimensioni di ***6 metri*** e di ***4 metri***. Quanto spenderà se sceglie quella che costa 5 euro al metro lineare?

PROCEDIMENTI E DATI

La conoscenza dei procedimenti e quella dei dati nella mente dell'uomo

I procedimenti non servono solo a risolvere i problemi di matematica, come abbiamo visto nella fase precedente. Nella realtà quotidiana l'andare da casa e scuola o a trovare gli amici, lo svolgere in palestra un certo esercizio di ginnastica, scartare un giocatore avversario e tirare in porta, sono procedimenti. Come abbiamo visto nei problemi generalizzati, anche in questi casi noi conosciamo i **procedimenti** ma, al momento dell'esecuzione, dovremo procurarci i **dati** necessari per la loro corretta esecuzione.

*Ad esempio, nel procedimento che comprendere tutte le azioni da compiere per **recarci da casa a scuola** vi sarà anche, nel momento in cui attraversiamo la strada, l'osservazione della posizione delle automobili presenti in quel momento sulla strada oppure, se attraversiamo un incrocio, l'osservazione del colore del semaforo.*

Anche in questo caso noi abbiamo memorizzato dei procedimenti e dobbiamo, ogni volta che li eseguiamo, procurarci i dati necessari per svolgerli.

Tutti i procedimenti che una persona conosce e di cui ha ***padronanza**** vanno a costituire la **conoscenza dei procedimenti** *(**conoscenza procedurale**)* di quella persona.

padronanza > capacità di svolgere procedimenti velocemente e senza pensarci sopra.

Si definisce invece con il termine di **conoscenza dei dati** tutto ciò che quella persona sa e che non fa parte della conoscenza procedurale:

ad esempio cognome e nome dei nostri compagni di classe, degli insegnanti, dei giocatori di calcio preferiti, di città, stati, ecc…

Come abbiamo visto nei problemi generalizzati, anche i procedimenti che noi svolgiamo quotidianamente richiedono, per essere svolti correttamente, la conoscenza di dati.

Ad esempio:

- *anche se conosciamo il procedimento per fare le divisioni, dovremo comunque prima memorizzare i numeri da dividere.*
- *anche se conosciamo il procedimento per recarci da casa a scuola, ogni volta che attraversiamo la strada dovremo controllare se arrivano auto e attraversare solo in caso di risposta negativa*
- *anche se conosciamo il procedimento per tirare in porta, prima di tirare dovremo osservare bene la posizione della palla, del portiere, ecc…*

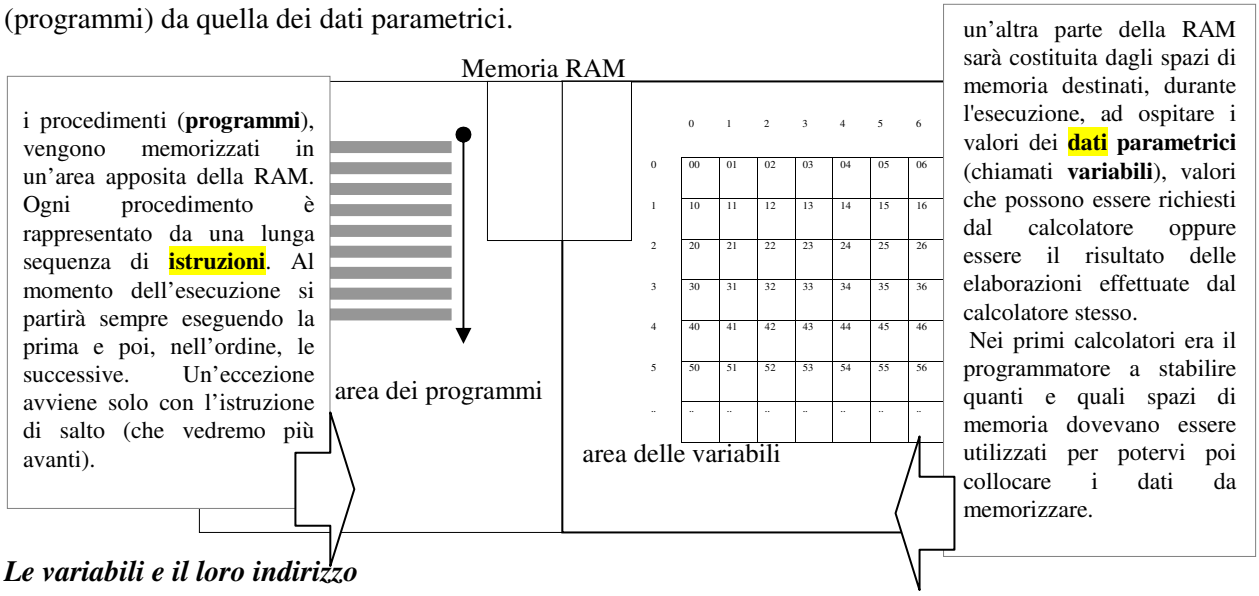
Completa il seguente esercizio scrivendo i nomi di sei procedimenti che fanno parte della tua conoscenza procedurale e i nomi di sei dati che ti devi procurare durante il loro svolgimento

procedure	*dati*
fare le divisioni	*numeri da dividere*
attraversare la strada	*colore del semaforo*
tirare il pallone in porta	*posizione del portiere, della palla, dei giocatori*

Mentre il procedimento è stabilmente memorizzato nella parte di memoria che ospita la conoscenza procedurale e potrà essere eseguito senza pensarci sopra, i valori dei dati dovranno essere ospitati in un'altra parte di memoria.

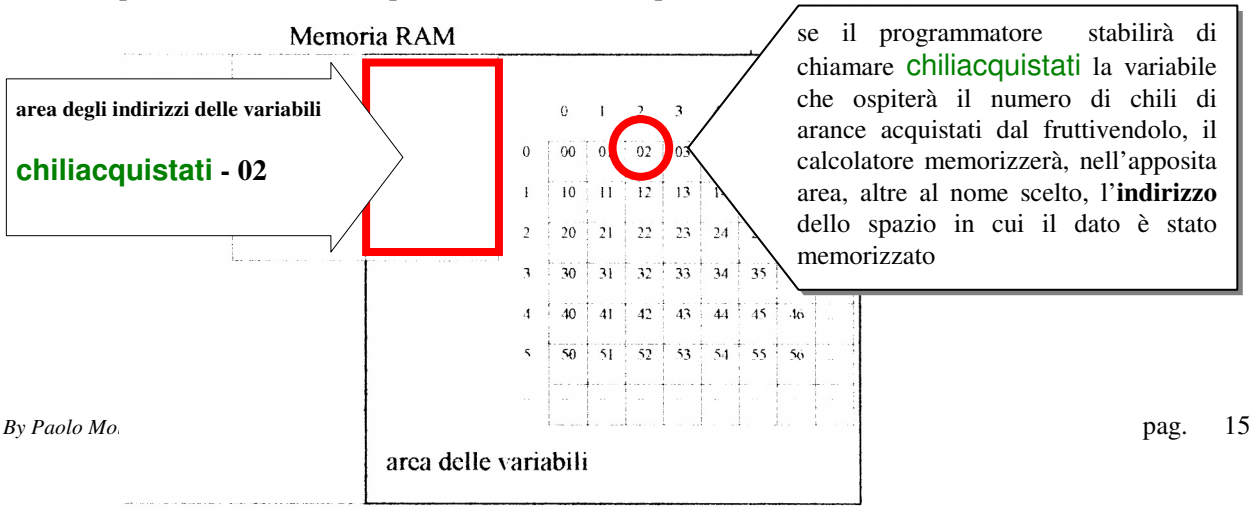
La "conoscenza" dei procedimenti e quella dei dati nella RAM del calcolatore
l'area dei programmi e l'area delle variabili

La mente dell'uomo è fatta di materiale organico che, durante l'apprendimento, si modifica stabilendo legami di significato tra le informazioni che man mano vengono memorizzate. Dobbiamo invece immaginare la memoria di un calcolatore come una cassettiera con tanti cassetti numerati nei quali vengono collocate le informazioni. Ogni informazione è raggiungibile solo conoscendo il numero del cassetto (**indirizzo di memoria**) in cui è stata collocata. Il calcolatore non è in grado di stabilire collegamenti tra gli spazi di memoria che ospitano i dati e dunque, autonomamente, non è in grado di ricercare dei dati partendo da altri dati che hanno con essi legami di significato.
Questo fatto ha importanti ripercussioni anche sui linguaggi che noi dobbiamo usare per comunicare con esso. Mentre nei linguaggi naturali è possibile risalire alle differenze di significato di segni simili, oppure interpretare frasi identiche in modo diverso proprio risalendo al contesto nel quale sono inserite, la comunicazione con il calcolatore non potrà mai essere ambigua, ad ogni segno deve corrispondere un solo significato.
Anche il calcolatore, come la mente dell'uomo, gestisce in modo diverso la conoscenza procedurale (programmi) da quella dei dati parametrici.



Le variabili e il loro indirizzo

Oggi non è più necessario che il programmatore ricordi gli indirizzi degli spazi di memoria liberi, questo lavoro viene svolto dal calcolatore. Il programmatore dovrà dare un nome (**nome della variabile**) allo spazio di memoria nel quale vuole memorizzare un dato. Il calcolatore ne cercherà uno libero e vi inserirà il dato. In un'apposita area della RAM (**area degli indirizzi delle variabili**) il calcolatore memorizza il nome scelto dal programmatore affiancato dall'indirizzo dello spazio di memoria. Ogni volta che nel programma verrà citato quel nome il calcolatore ne cercherà l'indirizzo per poi raggiungere il dato e compiere l'operazione comandata.
A seconda di che cosa si vuole memorizzare (numeri interi, numeri decimali, caratteri, date, ecc …) è possibile utilizzare diversi tipi di variabili distinguibili tra di loro dalla eventuale presenza, nel nome, di caratteri speciali. Per ora ci occupiamo di variabili che possono contenere numeri.



i nomi delle variabili

I nomi delle variabili possono essere rappresentati da una successione di lettere e di numeri che però non possono essere separati da spazi bianchi e non possono iniziare con un numero. Vanno inoltre evitati caratteri speciali (es: €). Inoltre i nomi non possono corrispondere ad istruzioni del linguaggio che si sta usando. Liberty Basic ci aiuta a non fare errori.
Infatti quando una linea è completata il calcolatore analizza e riconosce ciò che è stato scritto. Gli ordini vengono colorati in blu mentre i nomi delle variabili vengono colorati in verde muschio.
Nella scrittura è importante ricordare che il carattere maiuscolo è diverso da quello minuscolo.
Ad esempio la variabile guadagno è diversa dalla variabile Guadagno.

Il numero dei caratteri che possono formare un nome subisce dei limiti che variano a seconda del tipo di calcolatore e del linguaggio utilizzato (40 con il GWBASIC, ben 255 in JBASIC).

esercitazioni sulle variabili

Lavorando in diretta sul calcolatore potremo ora fare delle assegnazioni di variabili, cioè ordinare al calcolatore di organizzare nella sua memoria delle variabili e inserire in esse il valore di un dato.
Se ad esempio batterò sulla tastiera:

a = 125 seguito da ⏎

il calcolatore organizzerà una variabile chiamata **A** nella quale viene collocato il numero 125. Questo valore potrà essere utilizzato anche in operazioni successive.
Se batterò sulla tastiera:

b = 2*a seguito da ⏎

viene organizzata una nuova variabile chiamata **B** nella quale viene collocato il risultato dell'operazione 2 * 125 (quest'ultimo valore prelevato dalla variabile A). Questo risultato (250) apparirà sullo schermo battendo:

print b seguito da ⏎

Di conseguenza quando in una elaborazione matematica vi è il nome di una variabile il calcolatore andrà sempre a leggerne il valore e lo utilizzerà per il calcolo.

in BASIC non è obbligatorio assegnare una variabile prima di poterla usare, ad esempio in un'elaborazione. Quando il calcolatore trova il nome di una variabile non precedentemente assegnata gli assegna automaticamente il valore zero. Infatti battendo:

print k seguito da ⏎

il calcolatore scriverà sullo schermo il numero zero.
Questo fatto, come abbiamo già visto, può causare problemi in caso di errori di scrittura.

Potremo ora esercitarci ad organizzare nuove variabili nel calcolatore assegnando loro dei valori e ad utilizzarle per compiere elaborazioni.

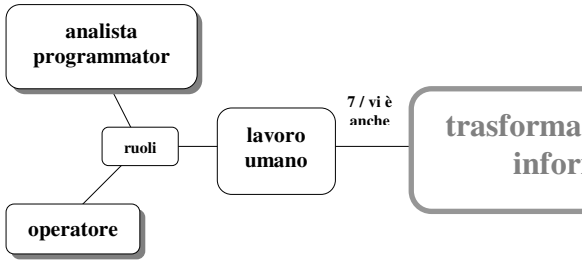
Il lavoro umano

Come abbiamo visto, la presenza di dati con valore costante restringe le possibilità di utilizzo del procedimento. Sarà compito dell'**analista** dunque "parametrizzare" il testo del problema, trasformare cioè, quando possibile, il maggior numero di dati costanti in dati parametrici, prevedendo che il calcolatore, all'inizio dell'esecuzione, ne chieda ed ottenga i valori.

Essi verranno forniti dall'**operatore**, cioè la persona presente davanti alla tastiera al momento dell'esecuzione. L'elaborazione automatica delle informazioni richiede anche un'organizzazione del lavoro umano:

l'**analista** è la persona che studia come risolvere un determinato problema con il calcolatore, mentre il **programmatore** è colui che scrive le istruzioni in linguaggio artificiale. Spesso questi due ruoli sono riuniti nella stessa persona.

l'**operatore** è la persona presente davanti alla tastiera al momento dell'esecuzione. E' perciò colui che gli fornirà i valori dei dati parametrici e che riceverà il risultato finale



in sintesi

In quale memoria del calcolatore e in quale sua area sono ospitati i dati parametrici? _____

A quali tipi di conoscenza dell'uomo sono simili? _____

Qual è la differenza sostanziale tra il modo con cui l'uomo memorizza le informazioni e quello utilizzato dal calcolatore? _____

_____

Quale organizzazione del lavoro prevede l'attività con il calcolatore?

_____

ed ora al lavoro su nuovi problemi

I seguenti problemi sono simili ad alcuni che hai già fatto, ma più complessi:

Problema specifico 5/a

Un'agenzia immobiliare ha comperato un appartamento di ***80 mq*** per ***20.000 euro***. Lo ristruttura spendendo in media ***100 euro*** al metro quadrato.
Se poi lo ha rivenduto a ***400 euro*** al metro quadrato, quanto ha guadagnato?

Problema specifico 5/b

Un'agenzia ha comperato un appartamento di ***90 mq*** per ***45.000 euro***. Lo ristruttura spendendo in media ***200 euro*** al metro quadrato per rifare il pavimento e altri ***4.000 euro*** per gli impianti (elettrico, idraulico, ecc..). Se lo ha poi rivenduto a ***900 euro*** al metro quadrato, quanto ha guadagnato?

Problema specifico 6/a

Un'impresa di pulizie paga un suo dipendente in base al numero di ore di lavoro effettuate. Le ore diurne vengono pagate ***10 euro*** mentre le ore effettuate di notte vengono pagate il ***doppio*** (dato costante).
Sapendo che nel corso del mese ha lavorato ***140 ore*** diurne e ***40 ore*** notturne calcolare la paga mensile del dipendente.

Problema specifico 6/b

Un'impresa di pulizie paga un suo dipendente con una paga base mensile di ***500 euro*** e con un aggiunta calcolata in base al numero di ore di lavoro effettuate. Le ore diurne vengono pagate ***5 euro*** mentre le ore effettuate di notte vengono pagate il ***doppio*** (da mantenere costante).
Sapendo che nel corso del mese ha lavorato **140 ore** diurne e **40 ore** notturne e che ***¼*** di quanto guadagnato (da mantenere costante) dovrà essere tolto per pagare le tasse, calcolare il guadagno effettivo del dipendente.

DATI ALFANUMERICI

nei primi calcolatori solo numeri

I calcolatori sono stati inventati per "fare calcoli" e dunque per elaborare numeri, opportunamente codificati in binario.

I numeri compresi tra 0 e 255 occupano un solo byte; per cifre superiori se ne dovevano aggiungere altri.

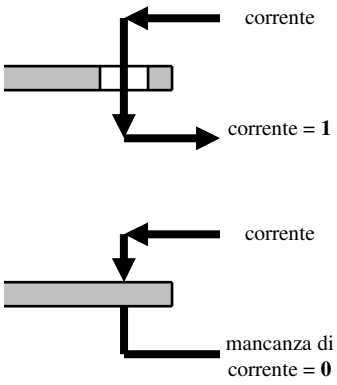
I primi calcolatori erano progettati per ricevere i dati da elaborare solo in formato binario (zero ed uno) e sempre in questo formato comunicavano i risultati delle elaborazioni. Questo avveniva grazie a delle schede rettangolari in cartoncino che venivano inserite in appositi *lettori*.

Il bordo delle schede riproduceva la situazione del byte da memorizzare. Un cerchio nella scheda indica la posizione dei singoli bit. Un foro all'interno del cerchio memorizza il numero uno, mentre la mancanza del foro memorizzava lo zero.

situazione della scheda prima della perforazione

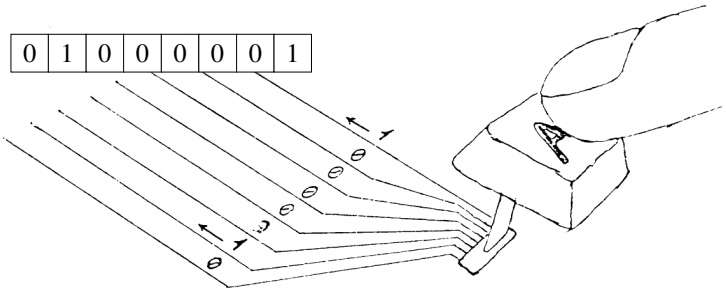
scheda perforata che riproduce il byte ***01010011***

Ovviamente, prima dell'inserimento nel lettore, la scheda doveva essere forata in modo da poter riprodurre i dati da memorizzare. Questa operazione veniva fatta manualmente da appositi addetti. Nel lettore delle apposite punte metalliche mobili, collegate ad un generatore di corrente, sono situate in corrispondenza dei cerchi della scheda. Se il cerchio è stato forato, la punta può infilarvisi e trasmettere la corrente; in caso contrario la corrente non potrà oltrepassare la scheda. In questo modo il byte memorizzato sulla scheda viene acquisito dal calcolatore diventando un blocco di otto fili, ognuno dei quali trasmette zero oppure uno a seconda se viene percorso dalla corrente o meno.



grazie ai <u>byte alfanumerici</u> compaiono i caratteri

Solo in un secondo momento, per comunicare con il calcolatore, le **tastiere** hanno sostituito le **schede perforate**. La pressione di un tasto della tastiera attiverà, in un fascio di otto fili (byte da 8 bit), il passaggio di correnti elettriche che percorreranno (o non percorreranno) i singoli fili a seconda del tasto che è stato premuto. Chi ha progettato la tastiera ha fatto sì che la pressione di un tasto provocasse negli otto fili ad esso collegato il passaggio di correnti che trasmettono il codice di quel carattere.

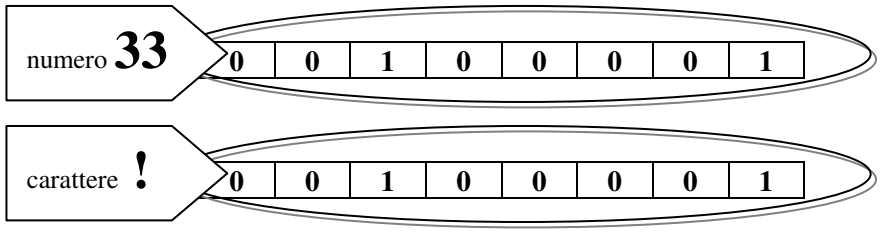


Per mettere il calcolatore in grado di scrivere messaggi sul monitor venne dunque studiata la possibilità di memorizzare nel calcolatore caratteri di testo, segni di punteggiatura, spazi e tutto ciò che è necessario per una corretta comunicazione.

Ogni carattere deve però essere trasformato in numero binario, in base ad un codice prestabilito, e memorizzato in un byte da 8 bit.

Ovviamente i byte che vengono utilizzati per questo scopo (byte alfanumerici) <u>devono essere contrassegnati in modo diverso</u> rispetto a quelli che contengono numeri da utilizzare per le elaborazioni matematiche (byte numerici). I dati in essi contenuti dovranno essere gestiti in modo completamente diverso.

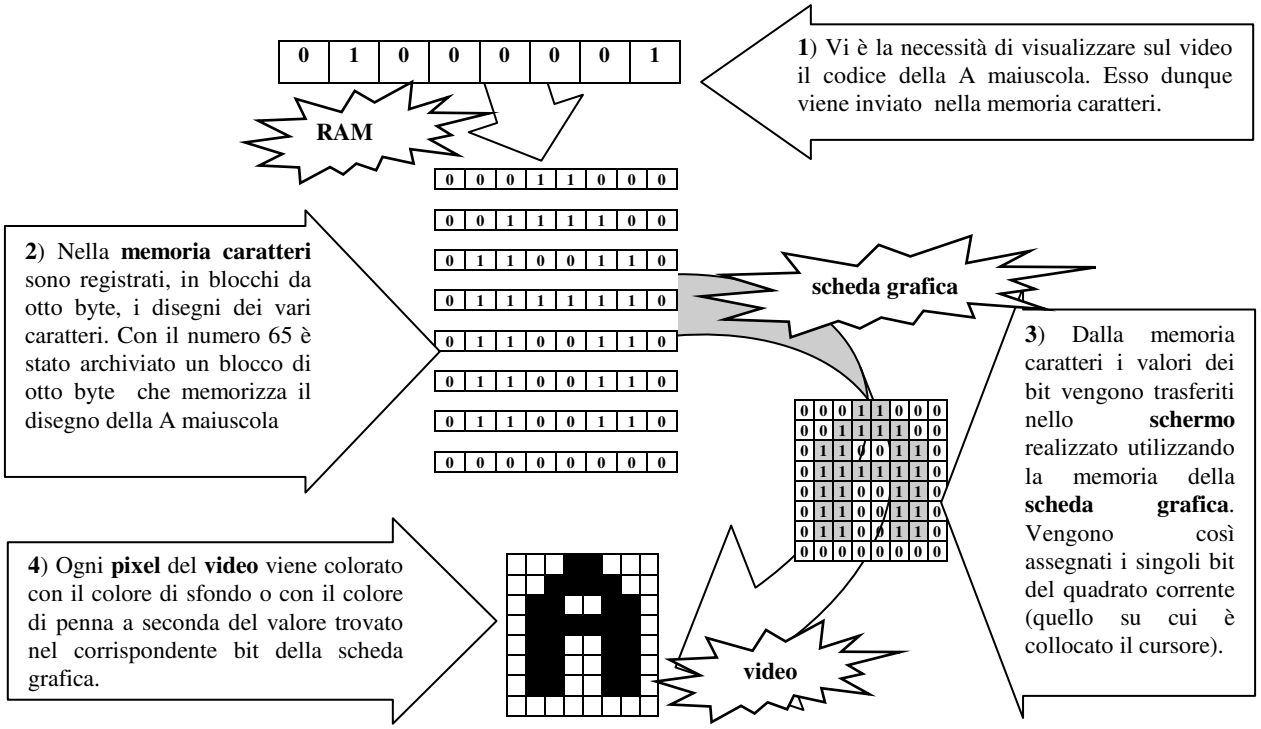
Infatti, nel nostro calcolatore, un byte che contiene il numero 00100001, se è numerico sta memorizzando il numero trentatre mentre se è alfanumerico, sta memorizzando il punto esclamativo (!).



i codici ASCII
Uno dei codici più diffusi, utilizzato anche dal nostro calcolatore, è il **codice ASCII**, cioè "American Standard Code for Information Excange", ovvero "codice standard americano per lo scambio dell'informazione". Nella tabella che segue sono riportati i vari caratteri e comandi in codice ASCII con i corrispondenti numeri decimali (da noi utilizzati per comunicare con il calcolatore) e i byte in codice binario utilizzati per rappresentare i caratteri all'interno del calcolatore.
Dal numero 32 al 126 sono codificati i caratteri normali, mentre da 0 a 31 e con il 127 sono codificati i "caratteri di controllo", codici che non corrispondono sempre alla stampa di un carattere, ma che possono avere anche un effetto particolare, come ad esempio andare a capo oppure generare un suono (con il codice 7) detto "bell", ovvero "campana".
I caratteri da 128 a 255 spesso variano significato a seconda della versione ASCII utilizzata.

l'uso della "memoria caratteri" per riottenere il disegno dei caratteri

Se all'interno del calcolatore, negli scambi con le periferiche (memorie secondarie, stampanti, ecc..) o negli scambi tra calcolatori (rete interna, Internet) il carattere viene sempre sostituito dal corrispondente byte alfanumerico, quando il calcolatore comunica con l'operatore (tramite monitor o stampa) vi è la necessità che il suo codice ASCII venga ritrasformato nel corrispondente carattere, da far comparire colorando i pixel sul monitor o sulla carta). Questo potrà accadere grazie alla presenza, nella RAM del calcolatore, della memoria caratteri, che viene memorizzata insieme alle istruzioni del sistema operativo.
Ad ogni codice di carattere corrisponde una griglia di byte che permettono di ottenere, il corrispondente disegno. L'esempio che vedremo in seguito riproduce il sistema utilizzato una trentina di anni fa con monitor in bianco e nero. Una griglia di otto byte è sufficiente per memorizzare il disegno di un carattere che, sul video, sarà ospitato da una corrispondente griglia di 64 pixel (piccoli quadratini in cui si divide il video. Ad ogni pixel corrisponde un bit. Il pixel sarà colorato di nero se il corrispondente bit conterrà 0, mentre sarà colorato di bianco se il bit conterrà 1 (in quei monitor il nero era il colore di sfondo mentre il bianco era il colore di penna). Però, prima di far comparire testi, grafici, disegni, sul video, il calcolatore compone le immagini in un'apposita area di memoria della scheda grafica.



Qui sarà realizzato un modello di ciò che si dovrà vedere, modello che non sarà formato da pixel luminosi, ma dai corrispondenti bit ospitanti lo zero o l'uno. D'ora in poi chiameremo schermo questa immagine virtuale formata da bit, mentre chiameremo video il luogo dove compare la corrispondente immagine luminosa. Questa distinzione è importante perché, come vedremo in seguito, un calcolatore nella sua scheda grafica può ospitare più schermi che però potranno essere visualizzati uno alla volta.
Una volta composta l'immagine, essa viene inviata verso il video, dove i bit provenienti dalla scheda grafica diventeranno quadratini più o meno luminosi.

E' importante tenere presente che l'attuale sistema operativo WINDOWS, integrato da potenti schede grafiche, permette al calcolatore di ottenere sullo schermo un numero molto elevato di pixel. E' perciò possibile disegnare i caratteri in diverse dimensioni. Il sistema operativo WINDOWS inoltre contiene varie *fonts* di caratteri che sono disegnati in differenti modi e utilizzano un maggior numero do pixel. Questi, a scelta, possono essere inseriti nella memoria caratteri della RAM.

* le diverse font riuniscono i caratteri dell'alfabeto disegnati con stili diversi

come conoscere i codici dei caratteri

Se vogliamo conoscere questo numero, tradotto in decimale, potremo utilizzare l'istruzione:

carattere
asc("_")
ad esempio con **print asc("B")** *otterrò sullo schermo il numero* ***66***
mentre con **print asc(a$)** *otterrò sullo schermo il codice del primo dei caratteri memorizzati dentro la variabile* ***a$***

Se vogliamo invece, dato un numero di codice, sapere a quale carattere corrisponde potremo usare l'istruzione:

numero
chr$(__)
Esempio: con **print chr$(67)** *otterrò sullo schermo il carattere* ***C***

i codici ASCII aprono nuove possibilità

La presenza dei **codici ASCII** ha offerto ai programmatori nuove e rilevanti possibilità, alcune delle quali saranno esaminate in questa fase.
Infatti i codici ASCII possono essere:

- utilizzati per ***scrivere le istruzioni del programma*** e per memorizzarlo provvisoriamente nella RAM in attesa della loro compilazione in linguaggio macchina (come avviene quando si programma in JBASIC)
- memorizzati, insieme al programma, per completare le ***comunicazioni rivolte all'operatore*** o per inserire ***annotazioni*** per lo stesso programmatore.
- memorizzati ***in particolari variabili*** ospitate nella RAM del calcolatore
- utilizzati ***per depositare su "memorie esterne"*** (dischetti, hard disk, cd, ecc…) non solo dati alfanumerici, ma anche dati numerici e programmi

nasce un linguaggio per gestire l'impaginazione dei caratteri: l'HTML

Dopo la scoperta del sistema per codificare i caratteri molti calcolatori iniziarono ad essere utilizzati per scrivere documenti di testo al posto delle macchine da scrivere (oramai sparite dalla circolazione). Furono messe a punto le prime applicazioni per gestire testi con il calcolatore (videoscrittura) e furono organizzate le prime banche dati elettroniche (archivi di dati memorizzati non più su carta ma su memorie secondarie di calcolatori).
Quando Internet (composto del latino inter, "fra" e dell'inglese net, "rete"), nata negli anni '60 per collegare tra di loro una serie di calcolatori presenti in basi militari, venne messa a disposizione dell'utenza civile si iniziò a studiare la possibilità di realizzare grandi archivi di dati consultabili in rete grazie ad appositi programmi.
Nel 1992 presso il CERN di Ginevra il ricercatore Tim Berners-Lee definì il protocollo HTTP (HyperText Transfer Protocol), un sistema che permette una lettura non-sequenziale dei documenti, saltando da un punto all'altro mediante l'utilizzo di rimandi (link o, più propriamente, hyperlink). Nasce così l'**HTML** (acronimo per Hyper Text Mark-Up Language) linguaggio usato per descrivere i documenti ipertestuali disponibili nel Web.

- tabelle dei codici ASCII

La tabella che segue indica quali sono i codici ASCII presenti nella memoria del nostro calcolatore. Buona parte dei codici iniziali corrispondono in realtà a delle istruzioni (ad esempio il codice 3 viene utilizzato per segnalare la fine di un testo -End of Text-). Essi vengono graficamente rappresentati con un quadratino. Anche se nella tabella compaiono i corrispondenti numeri decimali è bene ricordare che, nel calcolatore, i codici ASCII sono memorizzati utilizzando un byte binario (ad esempio A = 01000001).

1	45 -	89 Y	133 …	177 ±	221 Ý
2 ┐	46 .	90 Z	134 †	178 ²	222 Þ
3 └	47 /	91 [	135 ‡	179 ³	223 ß
4 ┘	48 0	92 \	136 ˆ	180 ´	224 à
5 │	49 1	93]	137 ‰	181 µ	225 á
6 ─	50 2	94 ^	138 Š	182 ¶	226 â
7 •	51 3	95 _	139 ‹	183 ·	227 ã
8 ◘	52 4	96 `	140 Œ	184 ¸	228 ä
9	53 5	97 a	141	185 ¹	229 å
10	54 6	98 b	142 Ž	186 º	230 æ
11	55 7	99 c	143	187 »	231 ç
12	56 8	100 d	144	188 ¼	232 è
13	57 9	101 e	145 ‘	189 ½	233 é
14	58 :	102 f	146 ’	190 ¾	234 ê
15 ☼	59 ;	103 g	147 “	191 ¿	235 ë
16 ┼	60 <	104 h	148 ”	192 À	236 ì
17 ◄	61 =	105 i	149 •	193 Á	237 í
18 ↕	62 >	106 j	150 –	194 Â	238 î
19 ‼	63 ?	107 k	151 —	195 Ã	239 ï
20 ¶	64 @	108 l	152 ˜	196 Ä	240 ð
21 ┴	65 A	109 m	153 ™	197 Å	241 ñ
22 ┬	66 B	110 n	154 š	198 Æ	242 ò
23 ┤	67 C	111 o	155 ›	199 Ç	243 ó
24 ↑	68 D	112 p	156 œ	200 È	244 ô
25 ├	69 E	113 q	157	201 É	245 õ
26 →	70 F	114 r	158 ž	202 Ê	246 ö
27 ←	71 G	115 s	159 Ÿ	203 Ë	247 ÷
28	72 H	116 t	160	204 Ì	248 ø
29	73 I	117 u	161 ¡	205 Í	249 ù
30 -	74 J	118 v	162 ¢	206 Î	250 ú
31	75 K	119 w	163 £	207 Ï	251 û
32	76 L	120 x	164 ¤	208 Đ	252 ü
33 !	77 M	121 y	165 ¥	209 Ñ	253 ý
34 "	78 N	122 z	166 ¦	210 Ò	254 þ
35 #	79 O	123 {	167 §	211 Ó	255 ÿ
36 $	80 P	124 |	168 ¨	212 Ô	
37 %	81 Q	125 }	169 ©	213 Õ	
38 &	82 R	126 ~	170 ª	214 Ö	
39 '	83 S	127	171 «	215 ×	
40 (	84 T	128 €	172 ¬	216 Ø	
41)	85 U	129	173 -	217 Ù	
42 *	86 V	130 ,	174 ®	218 Ú	
43 +	87 W	131 *f*	175 ¯	219 Û	
44 ,	88 X	132 „	176 °	220 Ü	

○ Dati alfanumerici nel programma

Durante l'esecuzione di un programma la richiesta dei singoli dati avviene facendo comparire nella finestra di esecuzione un punto interrogativo. Come abbiamo già potuto notare, quando i dati richiesti sono numerosi, si può creare confusione sull'ordine con cui i dati devono essere immessi. Inoltre, se l'**operatore** (cioè la persona che immette i dati nel momento dell'esecuzione) è una persona diversa da chi ha steso il programma (**programmatore**), è anche possibile che non si sappia che cosa sta chiedendo il calcolatore.
Lo schema seguente si riferisce alla situazione comunicativa che si crea tra programmatore e calcolatore durante l'attività di istruzione.

Il programmatore, nella stesura del programma, dovrà dunque farsi carico:
- delle istruzioni, in linguaggio artificiale, da dare al calcolatore,
- dei chiarimenti, da dare all'operatore durante l'esecuzione del programma, su che cosa chiede e su che cosa comunica il calcolatore,
- di scrivere annotazioni, che rimarranno scritte nel testo del programma, per se stesso o per altri programmatori che dovessero riesaminare il programma per apportarvi delle modifiche.

I ***chiarimenti per l'operatore*** vanno scritti tra virgolette all'interno delle istruzioni che gestiscono l'ingresso e l'uscita dei dati.
Ad esempio:

```
input "scrivi il prezzo di vendita delle arance"; prezzovendita
```

il testo viene scritto tra virgolette mentre, prima del nome della variabile, è obbligatorio il punto e virgola. Con:

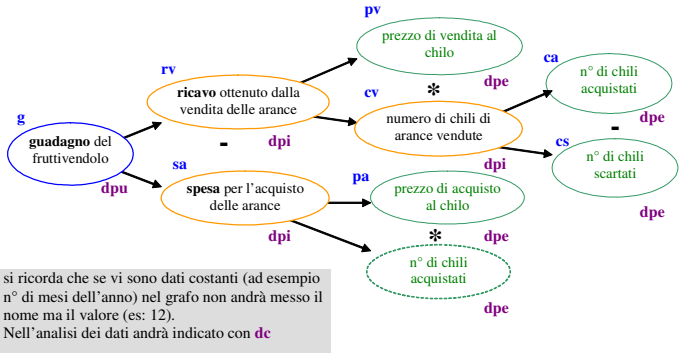
```
print "il guadagno del fruttivendolo è di ";guadagno;" euro"
```

la scritta "il guadagno del fruttivendolo è di " precede la scrittura del valore presente nella variabile guadagno, che poi è seguita dalla parola " euro" (da notare gli spazi in chiusura e in apertura delle virgolette). Come si può notare l'ordine print permette varie operazioni di scrittura successive. Visto che dopo una scrittura il calcolatore va a capo, il punto e virgola ordina che la scrittura successiva segua quella appena realizzata (con la virgola il calcolatore non va a capo ma si sposta di un certo numero di caratteri).
Sarà utile pulire la finestra di esecuzione prima della scrittura del risultato del problema utilizzando l'ordine **cls** (acronimo di ***cl**ear **s**creeen*).
Le ***annotazioni per il programmatore*** vanno precedute dall'ordine **rem** o, più comodamente, da un apostrofo. Ad esempio:

```
' programma per calcolare il guadagno di un fruttivendolo
```

Applichiamo le conoscenze appena apprese inserendo la parte relativa alla comunicazione nel programma precedentemente realizzato.

Problema specifico 0/b	Problema generalizzato 0/b
Un fruttivendolo acquista 100 kg. di aranci a 1 euro al chilo. Li rivende a 2 euro al chilo, avendone però scartati 10 kg. perché avariati. Quanto guadagna il fruttivendolo.	Un fruttivendolo acquista un certo numero di chili di arance ad un dato prezzo al chilo. Li rivende ad un prezzo al chilo maggiore dopo aver scartato un certo numero di chili di arance avariate. Quanto guadagna il fruttivendolo?



si ricorda che se vi sono dati costanti (ad esempio n° di mesi dell'anno) nel grafo non andrà messo il nome ma il valore (es: 12).
Nell'analisi dei dati andrà indicato con **dc**

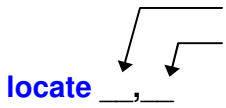
Per ottenere una valida comunicazione sarà importante anche la gestione grafica della finestra di esecuzione.
E' possibile collocare il cursore in una determinata posizione, pronto per la scrittura. Lavorando con il JBASIC, si utilizzerà:

procedura in linguaggio di progetto

```
leggi prezzo di vendita al chilo
leggi n° di chili acquistati
leggi n° di chili scartati
leggi prezzo di acquisto al chilo
ottieni numero di chili di arance vendute facendo
      n° di chili acquistati - n° di chili scartati
ottieni il ricavo ottenuto dalla vendita delle
      arance facendo prezzo di vendita al chilo *
      numero di chili di arance vendute
ottieni la spesa per l'acquisto delle arance
      facendo prezzo di acquisto al chilo * n° di
      chili acquistati
ottieni il guadagno del fruttivendolo facendo il
      ricavo ottenuto dalla vendita delle arance -
      la spesa per l'acquisto delle arance
scrivi il guadagno del fruttivendolo
```

programma

```
'programma fruttivendolo
'inizio ingresso dati
input "scrivi il prezzo di vendita delle arance (in €)";pv
input "scrivi il numero di chili acquistati ";ca
input "scrivi il numero di chili scartati ";cs
input "scrivi il prezzo di acq. delle arance (in €)";pa
'elaborazione
let cv = ca - cs
let rv = pv * cv
let sa = pa * ca
let g = rv – sa
cls 'pulizia e uscita dati
print "il guadagno del fruttivendolo è di "; g ;" €"
```



colonna su cui posizionare il cursore
riga su cui posizionare il cursore

Ad esempio: `locate 5,10:print "Laura"`
colloca il nome Laura sullo schermo. Il primo carattere sarà sulla colonna 5 e sulla riga 10

… ed ora al lavoro!

Qui di seguito vi sono alcuni problemi specifici. Dovrai trascriverne il testo sulle schede di programma date dall'insegnante e poi compiere tutti i passaggi sino ad ottenere il programma per il calcolatore.

Problema specifico 7/b

Un contadino ha nel suo orto ***5 alberi*** di arance e ricava da ognuno di essi ***10 chili*** di arance. Porta poi il raccolto al mercato e lo vende a ***2 euro*** al chilo. Per portarle al mercato viaggia per ½ ora (da mantenere costante) a ***80 chilometri*** orari con un furgone con un furgone che consuma ***0.5 euro*** al chilometro per la benzina. Quale sarà il guadagno del contadino?

Problema specifico 8

Un operaio deve lavorare ***40 ore*** alla settimana e con una paga oraria di ***9 euro***. Se ha fatto ***3 ore*** di straordinario e, complessivamente, ha percepito ***396 euro***, quanto riceve per ogni ora di straordinario?

Problema specifico 10

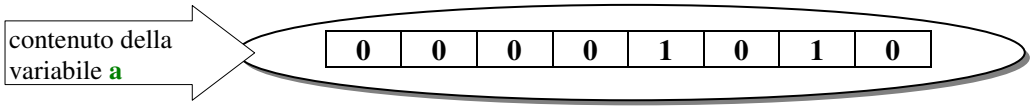
Un bimbo acquista ***3 pacchetti*** di caramelle, ciascuno dei quali ne contiene ***15***. Per strada mangia ***7 caramelle*** e, tornato a casa, regala a sua sorella metà (da mantenere come dato costante) delle caramelle rimaste. Quante caramelle restano al bimbo?

○ Le variabili alfanumeriche o "stringa"

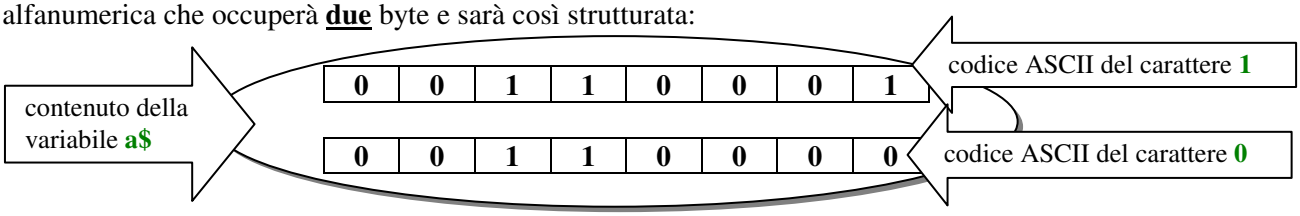
L'inserimento di testi dentro una variabile

I byte che codificano i caratteri presenti sulla tastiera, oltre che ad essere utilizzati nel programma per inserire commenti rivolti all'operatore o allo stesso programmatore, possono essere inseriti dentro delle apposite variabili. In questo caso la variabile si chiama **alfanumerica** o "stringa" e in quest'ultimo modo viene denominato anche il suo contenuto. Essa può contenere, in Liberty Basic, sino a 2 milioni di caratteri e viene distinta dalle variabili numeriche utilizzando il segno **$** di fianco al nome (ad esempio **a$**). Tale distinzione è necessaria visto il diverso trattamento che subiscono i due tipi di variabili.
I valori delle variabili numeriche vengono subito tradotti e archiviati in codice binario.

Se batterò **a = 10** seguito da ⊳ , il calcolatore organizzerà nella sua RAM una variabile numerica che occuperà **un** byte e sarà così strutturata:

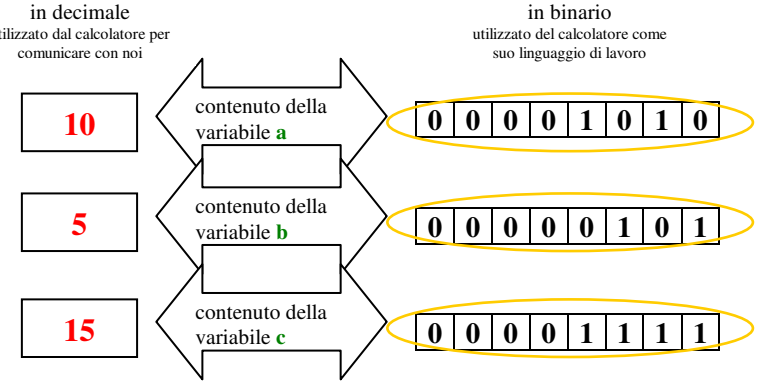


Mentre se batterò **a$ ="10"** seguito da ⊳, il calcolatore organizzerà nella sua RAM una variabile alfanumerica che occuperà **due** byte e sarà così strutturata:

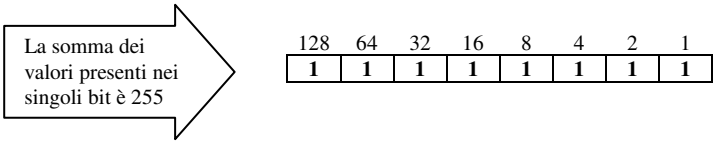


Di conseguenza:

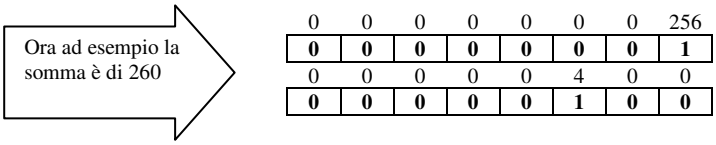
Se ad esempio ***a = 10*** *e* ***b = 5***
se assegno ***c = a + b***
dentro la variabile ***c*** *sarà collocato il numero* ***15****.*



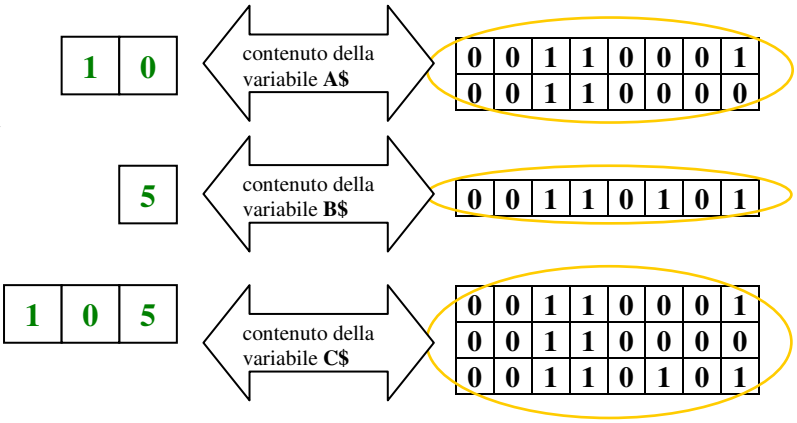
Infatti una variabile numerica basterà un byte per ospitare numeri sino al 255:



Per valori superiori bisognerà utilizzare i bit di un secondo byte



Se ad esempio ***a$="10"*** *e* ***b$="5"***
se assegno **c$=a$+b$** dentro la variabile **c$** sarà collocata la stringa **105**. Si leggerà unozerocinque e non centocinque in quanto non è stato ottenuto un numero, ma un insieme di caratteri**.**



Dunque l'organizzazione dei due tipi di variabili nella memoria del calcolatore avviene in modo completamente diverso. E' necessario distinguerle con un nome diverso.
Inoltre non potremo mai effettuare elaborazioni e controlli tra numeri e stringhe.

a$ + b NO!
c$ + 5? NO!
d = "Paolo"? NO!

Se assegniamo un insieme di caratteri ad una variabile alfanumerica (per esempio **A$="sedia"**) dovremo sempre chiuderlo tra virgolette; lo stesso si dovrà fare per qualsiasi parola o grafico che vogliamo scrivere o che vogliamo confrontare con il contenuto di una variabile alfanumerica.

Se è il calcolatore a chiederci il contenuto di una variabile alfanumerica tramite un'istruzione input, non si deve collocare l'informazione richiesta tra virgolette.

Nota Bene
In BASIC il calcolatore con le variabili numeriche considera contenenti **0** (zero) tutte le variabili non assegnate precedentemente, con le variabili alfanumeriche il calcolatore considera quelle non assegnate contenenti **""** (stinga nulla, da non confondere con " ").

spazio

L'uso di operatori algebrici e relazionali con le variabili "stringa"

Tra gli operatori algebrici solo il segno **=** (per le assegnazioni) e **+** (per unire il contenuto di più stringhe) possono essere usati con le variabili alfanumeriche.

Ad esempio: con a$="gioco" inserisco nella variabile a$ una parola (insieme di caratteri) che deve sempre essere tra virgolette.

nota lo spazio in coda

mentre se d$="montagna" e e$="rifugio di " con f$=e$+d$ metterò in f$ la frase "rifugio di montagna"

La gestione delle variabili alfanumeriche

Ogni carattere che memorizziamo dentro una variabile alfanumerica viene dunque ospitato in uno spazio di memoria (un byte).
Tali spazi saranno numerati in ordine crescente con un numero che indica la **posizione** di quel carattere all'interno della variabile.

Ad esempio dopo l'esecuzione dell'ordine:
a$="il gatto mangia il topo"
dentro la memoria del calcolatore la variabile A$ sarà così strutturata:

a$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
i	*l*		*g*	*a*	*t*	*t*	*o*		*m*	*a*	*n*	*g*	*i*	*a*		*i*	*l*		*t*	*o*	*p*	*o*

come si può vedere anche gli spazi bianchi vengono trattati come caratteri; lo stesso succede per tutti gli altri segni di punteggiatura.

Con l'istruzione:

mid$(__,__,__)

- nome della variabile alfanumerica
- posizione del carattere da cui partire nella lettura dei caratteri
- numero dei caratteri da leggere

E' possibile ottenere solo una parte del contenuto di una variabile alfanumerica (tale operazione viene chiamata **slicing.**

*Ad esempio: se utilizziamo la stringa collocata precedentemente in **a$** con* **mid$(a$,4,5)** *otterrò la parola "gatto".*

Mentre se non scriviamo il numero dei caratteri da leggere il calcolatore prosegue sino alla fine della stringa:

con **mid$(a$,17)** *otterrò la parola "il topo".*

Quest'ordine rende possibile riassegnare solo una parte di una variabile stringa.

Ad esempio: con **mid$(a$,4,5) = "micio"** *sostituirò la parola "gatto" con "micio".*

Per ottenere parte del contenuto di una variabile stringa sono disponibili anche le seguenti istruzioni:

left$(__,__)

- nome della variabile alfanumerica
- numero dei caratteri da leggere partendo dal primo carattere della stringa

Esempio: con **left$(a$,10)** *otterrà la parola "il gatto m"*

right$(__,__)

- nome della variabile alfanumerica
- numero dei caratteri da leggere partendo dall'ultimo carattere della stringa

Esempio: con **right$(a$,7)** *otterrà la parola "il topo"*

Inoltre l'istruzione:

len(__) stringa o nome dei una variabile stringa

fornisce il numero dei caratteri da cui è formata una stringa.

Esempio: con **print len(a$)** *il calcolatore scriverà sullo schermo il numero 23*

Possono essere molto utili anche i seguenti ordini:

lower$(__) stringa o nome dei una variabile stringa

converte in minuscoli gli eventuali caratteri maiuscoli presenti in una variabile alfanumerica.

Esempio: se a$="Quintino di Vona" con **print lower$(a$)** *il calcolatore scriverà "quintino di vona"*

stringa o nome dei una variabile stringa

upper$(__)

converte in maiuscoli gli eventuali caratteri minuscoli presenti in una variabile alfanumerica.

Esempio: se a$="Quintino di Vona" con **print upper$(a$)** *il calcolatore scriverà "QUINTINO DI VONA"*

stringa o nome dei una variabile stringa

trim$(__)

rimuove gli eventuali spazi presenti all'inizio o in coda di un testo.

Esempio: se a$=" Quintino di Vona " con **print trim$(a$)** *il calcolatore scriverà "Quintino di Vona"*

Scambi tra numerico e alfanumerico

Come sappiamo è possibile memorizzare dentro una variabile stringa un numero, ma non si possono effettuare con esso operazioni aritmetiche.
Per ovviare a questo inconveniente si può usare l'istruzione:

stringa o nome dei una variabile stringa

val(__)

Essa permette di utilizzare per operazioni matematiche il numero contenuto in una variabile stringa.
Ad esempio se: **b$="4"** con **a=2*val(b$)** potrò utilizzare quel carattere come numero per fare una moltiplicazione.

Esiste inoltre la possibilità di eseguire l'operazione inversa utilizzando l'istruzione:

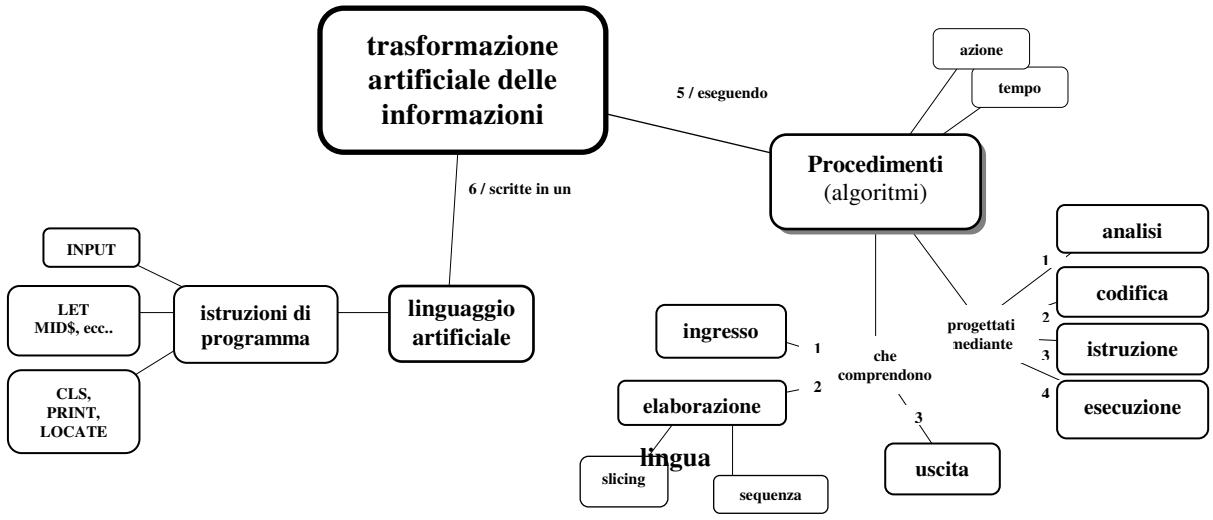
nome o contenuto di una variabile numerica

str$(__)

si può depositare il contenuto di una variabile numerica dentro una variabile stringa.

Esempi: **a=15:b$=str$(a)** *o* **b$=str$(15)** *o anche* **c$=str$(5)+str$(a)**

… in sintesi



Di quali argomenti ci siamo occupati in questa fase? _____

_____

Riepiloga qui sotto quanto riportato nella mappa utilizzando il sistema di lettura delle mappe che tu già conosci. Dove mancano le parole-legame userai quelle che ti sembrano più appropriate.

La trasformazione artificiale delle informazioni avviene eseguendo _____

_____

_____

_____

_____

_____

_____

… ed ora al lavoro!

Ora lavoreremo in diretta assegnando dei valori ad alcune variabili alfanumeriche e creando nuove variabili che contengano parole e frasi copiate dalle variabili precedenti tramite lo slicing. Controlleremo l'esattezza del nostro lavoro utilizzando l'ordine print.

Ad esempio se in **a$** inserisco:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
A	**l**	**d**	**o**		**l**	**a**	**v**	**o**	**r**	**a**		**c**	**o**	**n**		**i**	**l**		**c**	**o**	**m**	**p**	**u**	**t**	**e**	**r**

E in **b$** inserisco:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
i		**r**	**a**	**g**	**a**	**z**	**z**	**i**		**g**	**i**	**o**	**c**	**a**	**n**	**o**		**a**		**p**	**a**	**l**	**l**	**a**

con l'ordine:

```
c$=left$(b$,18)+right$(a$,15)
```

otterrò la seguente variabile:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
i		**r**	**a**	**g**	**a**	**z**	**z**	**i**		**g**	**i**	**o**	**c**	**a**	**n**	**o**		**c**	**o**	**n**		**i**	**l**		**c**	**o**	**m**	**p**	**u**	**t**	**e**	**r**

Inserendo in a$ la frase "quanto è bello giocare?" e in b$ la frase "raggiungere la meta è stato impegnativo" con quale ordine è possibile organizzare in c$ la frase "è bello raggiungere la meta"

c$= _____

e, utilizzando le stesse frasi, con quale ordine è possibile organizzare in d$ la frase "giocare è impegnativo?"

d$= _____

Inoltre inserendo in a$ la frase "Laura gioca a tennis" e in b$ la frase "Andrea non ama studiare" con quale ordine è possibile organizzare in c$ la frase "Andrea ama Laura"

c$= _____

e, utilizzando le stesse frasi, con quale ordine è possibile organizzare in d$ la frase "Andrea non gioca a tennis"

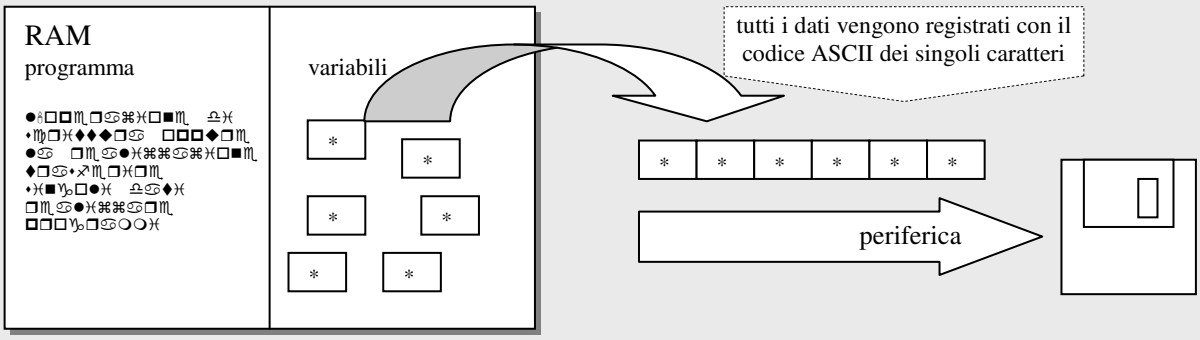
d$= _____

○ L'invio di dati verso le memorie secondarie

Salvare i dati presenti nell'area variabili della RAM

Come sappiamo, quando il calcolatore viene spento tutto ciò che è presente nella sua memoria RAM viene perso. Può essere dunque necessario copiare su una memoria secondaria (disco rigido, dischetto, nastro, CD, DVD, "penna", ecc…), oltre ai programmi realizzati, anche i dati ospitati nell'area variabili (acquisiti al momento dell'ingresso di dati o frutto dell'attività di elaborazione del calcolatore) al fine di poterli riutilizzare nuovamente in caso di necessità.
Mentre salvare il programma realizzato è un'operazione interamente gestita dal calcolatore (ordine SAVE), il salvataggio dei dati contenuti nella RAM è un'operazione un po' più complessa.

Quando vengono inviati verso una memoria secondaria i dati saranno collocati sequenzialmente uno dietro l'altro e in tal modo saranno registrati sul disco, divisi tra di loro da un apposito carattere di separazione(*). Essi vengono depositati, uno dopo l'altro, sul supporto scelto e saranno letti dal supporto alla stessa maniera.
Tutti i dati, quelli provenienti da variabili numeriche e da variabili alfanumeriche ma anche valori di dati costanti, vengono registrati su disco, carattere per carattere, con il proprio codice ASCII. Nel momento della lettura i numeri potranno essere depositati sul tipo di variabile che il programmatore riterrà utile.



per copiare i dati presenti nella RAM su una memoria secondaria sarà necessario realizzare un apposito **programma di gestione**, sotto la cui "regia" avverrà l'intera operazione

* lo spazio di memoria che ospita un singolo dato viene chiamato **campo**, mentre l'apposito codice utilizzato per segnalare il passaggio da un campo a quello successivo è il **separatore di campo.**
Come vedremo meglio in seguito, è possibile utilizzare come segno di separazione di campo i codici ASCII dell'***invio a capo*** (31) e della ***virgola*** (44).

In questa fase impareremo a realizzare programmi per trasferire singoli dati verso il disco rigido (hard disk) o un dischetto.
La realizzazione di questo trasferimento richiede la realizzazione di diverse operazioni:

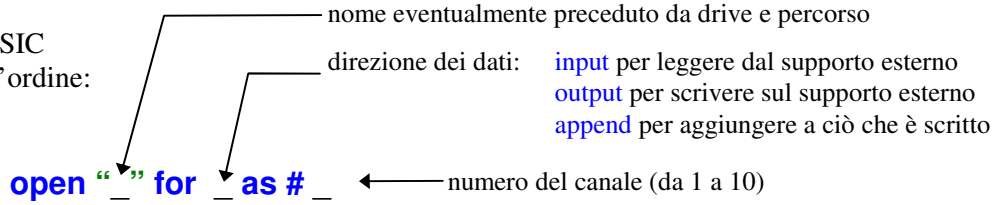
- l'operazione di apertura
- l'operazione di scrittura oppure l'operazione di lettura
- l'operazione di chiusura.

l'operazione di apertura

E' possibile aprire contemporaneamente più collegamenti verso le periferiche. Ad esempio sarà possibile leggere dati da un compact disk e memorizzarne altri sul disco rigido o su un dischetto.
E' dunque necessario "avvisare" il calcolatore che si intende inviare dei dati presenti nella sua memoria RAM verso una memoria secondaria sulla quale saranno collocati sequenzialmente, oppure che si intende leggere una sequenza di dati da una memoria secondaria per collocarli dentro delle variabili organizzate nella memoria RAM del calcolatore.
Verrà utilizzato un ordine di apertura che dovrà contenere le seguenti informazioni:

- la direzione dei dati; dalla memoria centrale verso il supporto (uscita) o viceversa (ingresso)
- il numero del canale (un numero convenzionale dato per differenziare tra loro i diversi collegamenti che possono essere gestiti contemporaneamente)
- il supporto verso il quale vengono indirizzati i dati qualora esso sia diverso da quello operativo
- il nome con il quale saranno registrati i dati oppure, in caso di lettura, con il quale sono stati memorizzati in precedenza. Esso potrà essere preceduto dall'eventuale percorso

In JBASIC si usa l'ordine:



esempio: con open "a:\alunni\92" for input as#1
viene aperto per la lettura un file chiamato 92 e collocato sulla directory alunni del disco collocato nel drive a. Come numero del canale viene scelto il 1.

Quando riceve l'ordine di apertura per la scrittura (output) il calcolatore controlla se il supporto richiesto è operativo. Verifica inoltre se vi sono ancora blocchi liberi e sposta la scrittura sul primo blocco libero. Se vi è già un file registrato con quel nome il calcolatore si prepara a cancellarlo per sostituirlo con la nuova registrazione.

Quando riceve l'ordine di apertura per la lettura (input) il calcolatore controlla se il supporto richiesto è operativo. Verifica inoltre se è presente un file registrato con il nome richiesto. In caso di risposta positiva sposta la lettura al blocco contenente il file richiesto preparandosi alla lettura.

Se una delle verifiche previste nell'ordine di apertura ha dato esito negativo il calcolatore dà un segnale di errore e non prosegue nelle operazioni.

E' importante tenere presente che il nome del file da aprire può essere contenuto dentro una variabile precedentemente assegnata.

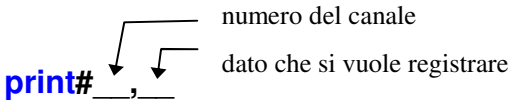
Ad esempio se: **c$="alunni"** *con* **open c$ for input as#1** *apriremo, per leggere, un file di nome alunni sul drive corrente*
Se prima del nome del file dobbiamo collocare drive e/o percorso tra virgolette dovremo agganciare ad esso il nome della variabile utilizzando il segno + (vedi *L'uso di operatori algebrici e relazionali con le variabili "stringa"*)

open "a:"+c$ for input as#1

Inoltre se il nome del file da aprire proviene da una variabile numerica è importante tener conto di quanto detto in *scambi tra numerico e alfanumerico*

la scrittura sul supporto

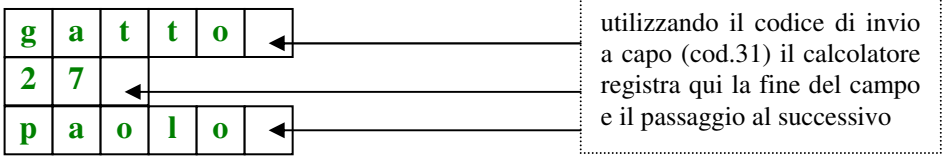
Per scrivere dati sul supporto si usano gli ordini:



che scrive i valori di tutto ciò che trova dopo la virgola sino all'invio a capo
se ad esempio a$ = "gatto": b = 27

```
print#1,a$
print#1,b
print#1,"paolo"
```

i dati saranno registrati nel seguente modo:



Mentre se scrivo:

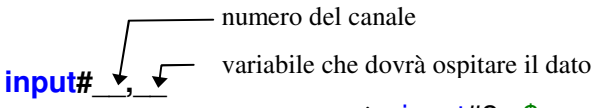
```
print#2,a$;b; "paolo"
```

i dati saranno registrati nel seguente modo:

Come si vede l'uso del punto e virgola riunisce tutti i dati all'interno dello stesso campo.

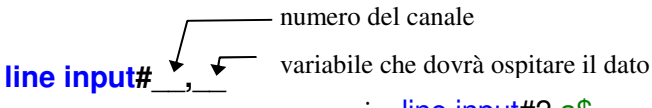
la lettura dal supporto

Per riportare nella RAM i dati salvati sulla memoria esterna potranno essere utilizzati gli ordini:



esempio: input#2,a$

Il calcolatore legge ciò che è collocato sul file sino al primo segno di separazione di campo: sia il codice dell'**invio a capo** (31) che il codice della **virgola** (44). Ciò che è stato letto sarà collocato dentro la variabile indicata dopo la virgola. Se il dato è un numero potrà essere collocato sia dentro una variabile numerica che alfanumerica mentre il dato alfanumerico potrà essere collocato solo dentro una variabile alfanumerica. In caso contrario il calcolatore darà un segnale di errore. Se nel file vi sono più campi sarà necessario effettuare più letture. In questo caso è possibile effettuare più letture con un solo INPUT# collocando più nomi di variabili separate da una virgola.
Ad esempio con: input#2,a$,b$,c$ il calcolatore legge in successione tre campi dal file e colloca i dati letti dentro le tre variabili indicate.



esempio: line input#2,a$

Il calcolatore legge ciò che è collocato sul file sino al primo **invio a capo** (cod.31). In questo caso anche le virgole vengono interpretate come normali caratteri e dunque il calcolatore leggerà e collocherà nella variabile tutto ciò che incontra sino al primo invio a capo.

Se, ad esempio, abbiamo registrato in un file di testo il noto titolo di film western:

i	l		b	u	o	n	o	,		i	l		b	r	u	t	t	o	,		i	l		c	a	t	t	i	v	o	

utilizzando **input#** sarebbero necessari tre ordini di lettura e il testo sarebbe diviso in tre variabili diverse, con **line input#** è necessaria una sola lettura e tutto il testo, sarà collocato in un'unica variabile.

Nota Bene > nella lettura del file, se si tenta di leggere più campi di quanti esso contenga, il calcolatore rimarrà bloccato da un segnale di errore alla fine dei campi disponibili.

l'operazione di chiusura

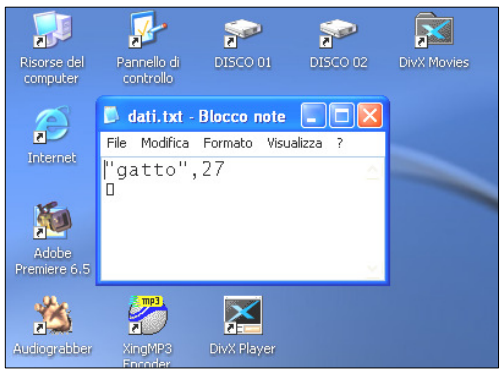
Dopo aver letto oppure scritto i dati bisognerà chiudere il file, operazione necessaria se si vuole poter utilizzare la periferica per altre operazioni.
L'ordine di chiusura è:

numero del canale

```
close# __
```

con la sua esecuzione viene chiuso il collegamento tra calcolatore e periferica.
Alla chiusura il calcolatore collocherà un apposito segno che viene chiamato segno di EOF (End Of File) corrispondente al codice ASCII n° 26.
se ad esempio:

```
a$ = "gatto":  b = 27
open ,"c: dati.txt" for output as#1
print#1,a$: print#1,",":print#1,b
close#1
```



Aprendo con Blocco Note il file di testo creato dopo i due dati memorizzati compare, alla riga successiva, il segno di EOF che, come abbiamo già visto dalla tabella dei codici ASCII, viene rappresentato con un quadratino.

Prendiamo il problema a fianco. Il programma che studieremo per risolverlo sarà utilizzato dall'impresa ogni volta che svolgerà dei lavori utilizzando la scavatrice.
Il calcolatore chiederà i seguenti dati:

numero di operai
n° di ore svolte da ogni operaio
costo orario di ogni operaio (in €)
n° di ore svolte dalla scavatrice
costo orario della scavatrice (in €)

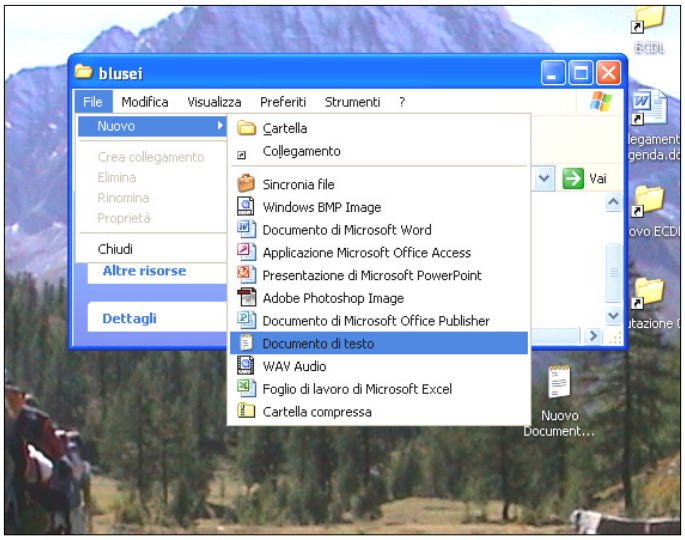
Problema specifico 9

Per effettuare degli scavi vengono impiegati 8 operai per 10 ore ciascuno e una scavatrice meccanica per 8 ore. Se ogni operaio percepisce una paga oraria di 10 euro e il costo orario della scavatrice è stato di 100 euro, quanto sono costati complessivamente i lavori?

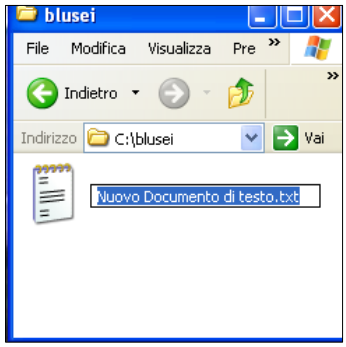
Problema generalizzato

Per effettuare degli scavi vengono impiegati un certo numero di operai per un certo numero di ore ciascuno e una scavatrice meccanica per un certo numero di ore. Se ogni operaio percepisce una paga oraria di un certo numero di euro e il costo orario della scavatrice è stato di un certo numero di euro, quanto sono costati complessivamente i lavori?

Invece di memorizzare, ad ogni lavoro, i dati da tastiera possiamo memorizzare i dati dei lavori su file di testo dando ad ogni file il nome del mese in cui sono stati effettuati i lavori. Realizzeremo dunque un programma in cui il calcolatore, dopo aver chiesto il nome del mese in cui sono stati svolti i lavori, leggerà i dati di quel mese, calcolerà il costo dei lavori e lo scriverà sul monitor.

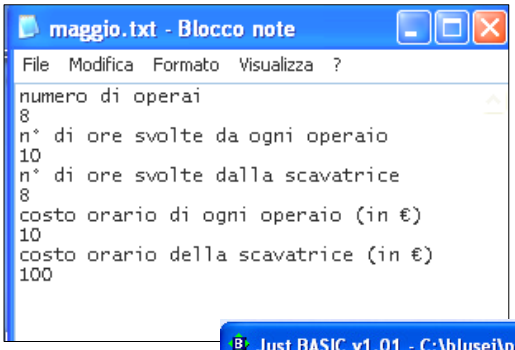


Per creare un primo file di dati apriamo la cartella **di gruppo** ed apriamo un file di testo

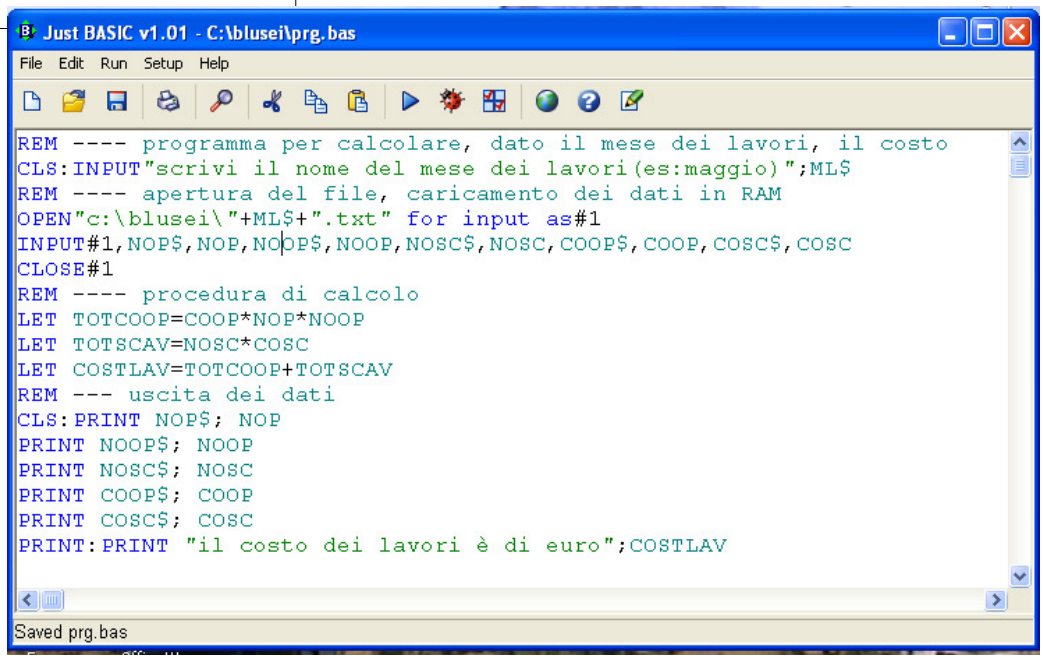


e dopo aver cancellato il nome provvisorio lo sostituiremo con il nome del mese scelto seguito da .txt

Inseriremo nel foglio di Blocco Note i dati del problema di matematica facendoli precedere dal nome del dato. L'invio a capo equivale al segno di separazione di campo. Dunque batteremo Invio per separare un dato da quello che dovrà essere letto successivamente.



Il file realizzato contiene dunque dieci dati (cinque nomi di dati seguiti dai rispettivi valori). Salveremo il file e, senza chiudere, potremo sostituire ai dati di maggio dei nuovi dati salvando il tutto con il nome di un altro mese. Con questo sistema potremo creare altri file.
Aperta poi la finestra di *Liberty Basic* inizieremo a scrivere il programma.



Proveremo poi il programma realizzato prima sul mese di maggio e poi sugli altri file di dati da noi archiviati nella ***cartella di gruppo***.

Aggiungiamo ora al programma le istruzioni per sostituire al file appena letto (dunque salvare con lo stesso nome) un file contenente solo il costo totale degli operai, il costo totale della scavatrice e il costo totale dei lavori effettuati in quel mese.

```
REM ---- programma per calcolare, dato il mese dei lavori, il costo
CLS:INPUT"scrivi il nome del mese dei lavori(es:maggio)";ML$
REM ---- apertura del file, caricamento dei dati in RAM
OPEN"c:\blusei\"+ML$+".txt" for input as#1
INPUT#1,NOP$,NOP,NOOP$,NOOP,NOSC$,NOSC,COOP$,COOP,COSC$,COSC
CLOSE#1
REM ---- procedura di calcolo
LET TOTCOOP=COOP*NOP*NOOP
LET TOTSCAV=NOSC*COSC
LET COSTLAV=TOTCOOP+TOTSCAV
REM --- uscita dei dati
CLS:PRINT NOP$; NOP
PRINT NOOP$; NOOP
PRINT NOSC$; NOSC
PRINT COOP$; COOP
PRINT COSC$; COSC
PRINT:PRINT "il costo dei lavori è di euro";COSTLAV
```

Controlliamo ora, utilizzando Blocco Note, il risultato della nostra sostituzione.

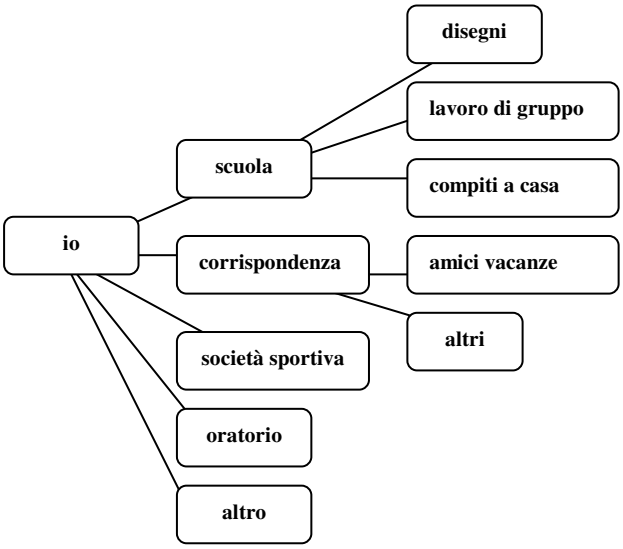
L'organizzazione dei file su disco

In genere a chi utilizza molto il calcolatore capita facilmente di produrre e salvare su hard disk grandi quantità di file contenenti testi, immagini, disegni, ecc… Se tutti i file fossero collocati insieme, andando a leggere il contenuto del disco, ci si troverebbe di fronte a lunghissimi elenchi, con la conseguente difficoltà ad individuare rapidamente ciò che ci interessa. E' perciò possibile, a somiglianza di quanto succede già nella nostra mente, organizzare delle strutture ad albero nelle quali i nodi (chiamati **cartelle** o **directory**) hanno la funzione di condurre ai file, stabilendo legami logici tra quelli che dipendono dallo stesso nodo. Mentre i file vengono utilizzati per depositare programmi e dati sul disco, le strutture ad albero servono per "organizzare" questi archivi, aggregandoli, in base ai dati contenuti e alle modalità di accesso, intorno a dei nodi.

Il disco appena formattato ha una sola cartella, che rappresenta la radice dell'albero. A partire da essa possiamo organizzare vari percorsi con nodi a cui collegare, successivamente, file.

Questa organizzazione, che va progettata con cura, ci permette anche di creare con facilità copie di sicurezza dei file che riteniamo importanti.

Di fianco vediamo una possibile organizzazione per cartelle destinate ad ospitare vostri file. Il fatto che tutte le cartelle con i file di produzione personale facciano capo ad un'unica (la cartella "io") rende semplice fare copie di sicurezza.



In WINDOWS la creazione di nuove cartelle, la loro gestione e lo spostamento dall'una all'altra possono avvenire molto facilmente utilizzando il mouse. Queste operazioni sono però possibili anche in DOS e in BASIC utilizzando i seguenti ordini:

nome di una variabile numerica

eventuale percorso e nome

```
_ = mkdir("_")
```

viene creata una nuova cartella (***dir**ectory* in inglese)

Ad esempio con `a= mkdir("c:gite\adda")`, *pur restando sulla cartella principale, il calcolatore organizza una nuova cartella chiamata **adda** all'interno della cartella **gite**. La variabile* `a` *permette al calcolatore di segnalare se l'operazione è riuscita. Infatti se scriviamo:*

```
a= mkdir("c:gite\adda")
print a
```

Il calcolatore scriverà il numero 0. Se viene scritto un numero diverso da 0 vuol dire che l'operazione non è andata a buon fine.

Per cancellare una cartella bisogna prima accertarsi che sia vuota e scrivere:

nome di una variabile numerica

eventuale percorso e nome

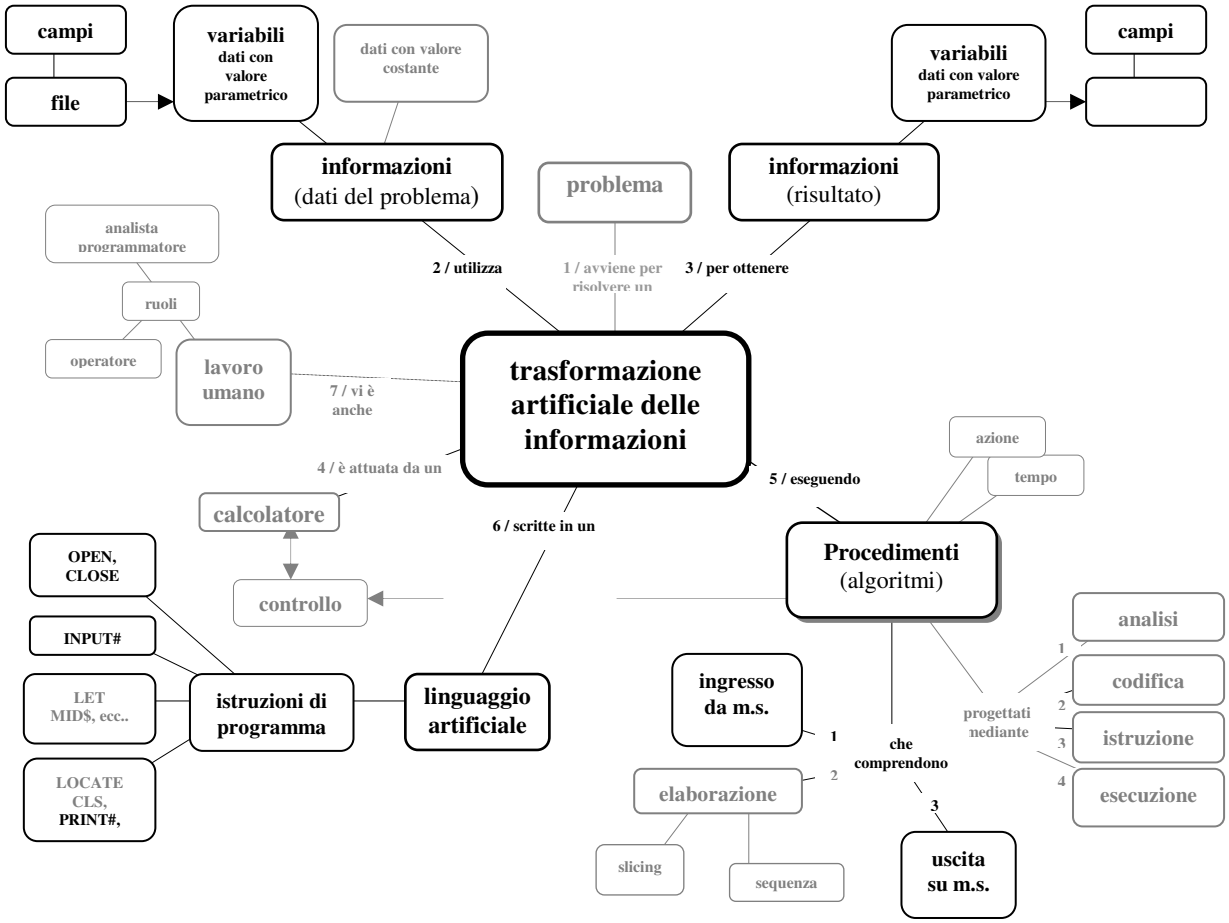
```
_ = rmdir("_")
```

La variabile numerica ha lo stesso compito visto con mkdir.

Per evitare di "perdersi" nei vari nodi del proprio dischetto finendo magari per salvare qualche file in posti diversi da quelli voluti è bene evitare di spostarsi dalla cartella principale. Per salvare o caricare i propri file o i propri programmi basterà far precedere i loro nomi dal percorso da effettuare.

Ad esempio con `open "c:alunni\3" for output as#1` *aprirà un file chiamato 3 sulla cartella **alunni** dell'hard disk (**c**)*

Ed ora al lavoro!



La mappa riassume tutto il percorso fatto evidenziando gli argomenti trattati in quest'ultima fase: il recupero o l'invio di dati tra memoria centrale e memorie secondarie e i procedimenti e istruzioni che rendono possibile queste operazioni.
Quando può essere utile registrare su disco i dati di ingresso utili per la risoluzione di un problema?

_____
_____

quando può essere utile registrare su disco i dati di uscita di un problema? _____
_____
_____

Cosa è un campo? _____

A che cosa serve il segno di separazione di campo? _____
_____

Quale carattere viene utilizzato come segno di separazione di campo? _____

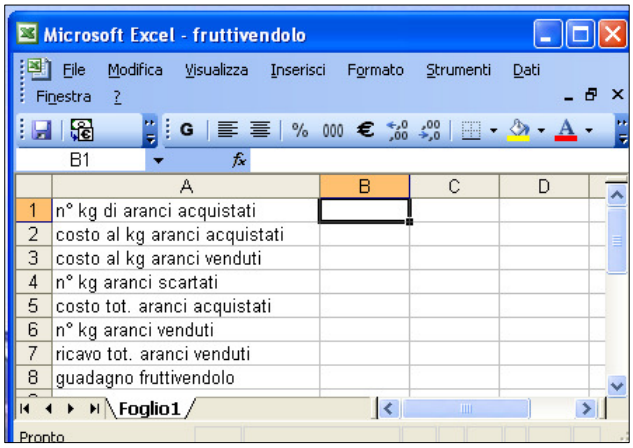
RISOLVERE I PROBLEMI CON I FOGLI ELETTRONICI

Sino ad ora abbiamo imparato a risolvere classi di problemi preparando noi stessi il software necessario per risolverli. Però, come sappiamo, la larga maggioranza delle persone che usa un calcolatore non è in grado di realizzare dei programmi. Se le aziende possono far produrre il software a loro necessario da appositi studi professionali, cosa può fare il singolo utente quando si trova di fronte ad una situazione problematica?

Come sappiamo varie società che producono software hanno realizzato delle applicazioni in grado di aiutare coloro che intendono utilizzare il calcolatore per scrivere dei testi (programmi di videoscrittura). La stessa cosa è successa per chi intende utilizzare il calcolatore per risolvere dei problemi. Le applicazioni studiate per questa necessità vengono chiamate **fogli elettronici** o fogli di calcolo.
Diamo per scontato che voi abbiate già imparato ad utilizzarli per realizzare semplici calcoli; vediamo ora come utilizzarli per risolvere non solo un singolo problema ma anche una classe di problemi (come un programma in BASIC).
Per fare questo riprendiamo in considerazione il problema del fruttivendolo e vediamo come può essere risolto utilizzando i fogli elettronici.

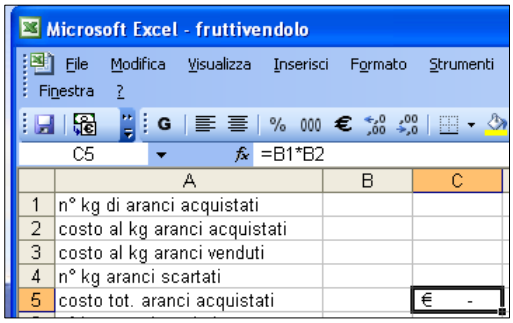
l'impostazione del foglio di lavoro

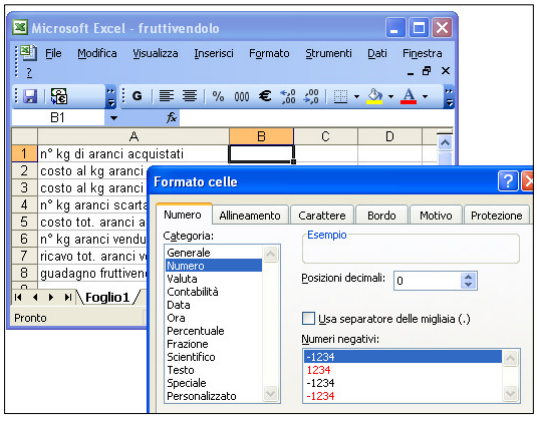
Apriamo un foglio elettronico (teniamo aperto solo il Foglio 1 eliminando gli altri due fogli di lavoro). Utilizziamo la colonna A, dopo averne formattate le celle come **testo**, per ospitare i nomi dei dati. Scriviamo prima i nomi dei dati d'ingresso, poi quelli che dovranno essere calcolati.
Per ultimo il dato di uscita (in questo caso il guadagno del fruttivendolo).



Per migliorare la leggibilità del Foglio è bene utilizzare la colonna B per scrivere i valori dei dati in ingresso. La colonna C ospiterà i risultati dei calcoli svolti dal computer e la colonna D il risultato finale.
Anche il nostro Foglio dovrà servire non per risolvere un problema ma una classe di problemi. Trascuriamo dunque, per il momento, che cosa scrivere sulla colonna B e passiamo subito ad esaminare che cosa scrivere sulla colonna C.

Il costo totale degli aranci acquistati sarà dato dal n° di kg di aranci acquistati (in **B1**) per il loro costo al kg (in **B2**). Selezionata la cella **C5**, scriviamo **=B1*B2** sulla barra della formula e formattiamo la cella nella categoria **valuta**. Subito compare il segno dell'euro che indica quale tipo di dati sarà ospitato da quella cella.

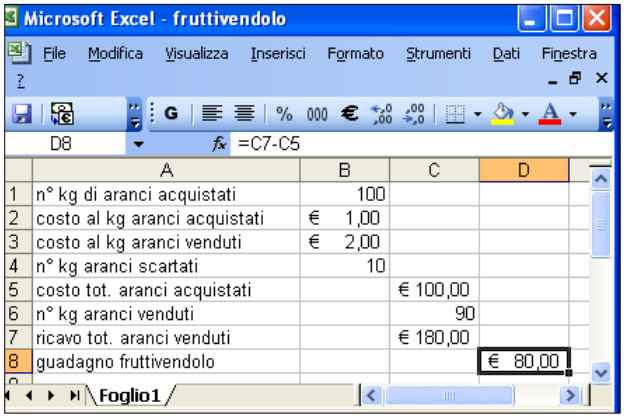




Poi, dopo aver formattato le rispettive celle, individuiamo e scriviamo le formule per le celle **C6**, **C7** e **D8**. Per le celle che accoglieranno numeri (n° kg aranci venduti) dovremo scegliere la categoria **numero** impostando a 0 le posizioni decimali (in genere la frutta viene acquistata e venduta a chili). Dovremo formattare anche le celle della colonna B in modo coerente con i dati che dovranno ospitare.

la risoluzione del problema

Infine scriviamo, nella colonna B, i valori dei dati utilizzando per ora quelli del problema iniziale.
Subito il computer calcola e inserisce i dati nelle colonne C e D dove compare il guadagno del fruttivendolo (80 euro).
Se cancelliamo dalla colonna B i vecchi dati e ne inseriamo di nuovi potremo osservare che, immediatamente, vengono aggiornati i dati nelle altre colonne.



Se cancelliamo nuovamente i dati sulla colonna B, potremo salvare il Foglio come **modello**. Potrà essere aperto e utilizzato ogni volta che sarà necessario risolvere un problema appartenente a quella **classe di problemi**.

verifica su un nuovo problema

Possiamo ora individuare altri problemi, già svolti nelle fasi precedenti, e svolgere autonomamente tutto il percorso, aprendo dei fogli elettronici e compilandoli, controllando poi la correttezza dei risultati forniti da calcolatore.

Come abbiamo potuto verificare, è molto semplice utilizzare i Fogli elettronici per risolvere problemi per i quali non sono disponibili applicazioni specifiche. Un'applicazione Basic, se ben fatta, è più semplice da utilizzare ma richiede una progettazione più onerosa che, in genere, viene svolta da professionisti. L'utilizzo dei Fogli elettronici è invece alla portata di qualsiasi utente.

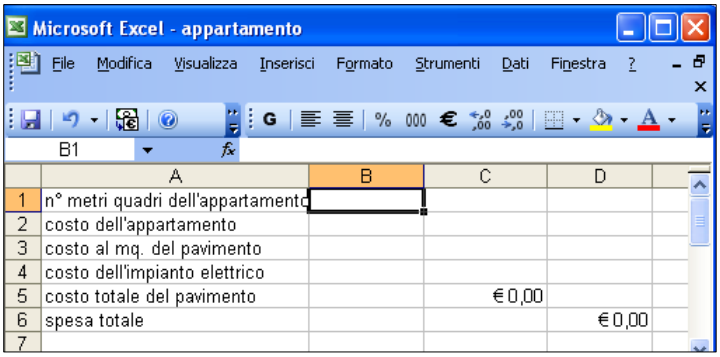
ed ora al lavoro!

Un foglio di lavoro viene salvato come modello perché: _____

Se una cella accoglierà dei prezzi è bene che sia formattata: _____

Se in una cella prevedo di utilizzare numeri interi formatterò la cella: _____

Per risolvere il seguente problema generalizzato:
Una famiglia ha acquistato un vecchio appartamento di un certo numero di metri quadri spendendo complessivamente una certa cifra. Per poterlo abitare devono essere sostituiti i pavimenti spendendo una certa cifra al metro quadro.
Altri soldi saranno spesi per rifare l'impianto elettrico. Calcolare quanto ha speso in tutto la famiglia per l'appartamento.



1) Che cosa deve essere scritto nella cella **C5**? =_____

2) Che cosa deve essere scritto nella cella **D6**? =_____

3) Se nella cella **B1** è stato scritto il numero **100** e nella cella **C5** compare **€ 10.000,00** vuol dire che nella cella **B3** è stato scritto: = _____ .

4) Inoltre se nella cella **B4** è stato scritto il numero **2000** e nella cella **B2** il numero **50000**, nella cella **D6** vi sarà: _____ .