

Unità di Apprendimento ALGORITMI (17/08/05)

area delle Informazioni

1 media

1. L'ambiente di lavoro:

Algoritmo > **codifica** > programma con ling. ad alto livello > **compilazione** > programma in linguaggio assoluto. Algoritmi **per fare calcoli** e algoritmi **per interagire con il mondo** (muoversi, vedere, ecc...)

1.a - i sistemi operativi

I sistemi operativi: MS-DOS e WINDOWS. Caricamento del DOS e del GWBASIC.

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

1.b - problemi e procedimenti

Il lavoro in diretta. Il problema e la classe di problemi. I dati con valore **costante** e quelli con valore **parametrico**. Il **problema di informatica** e quello di **matematica**.

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

1.b - la memorizzazione dei dati

La conoscenza procedurale e quella dichiarativa nella mente dell'uomo. L'area dei programmi e quella delle variabili nella RAM del calcolatore. L'**indirizzo** dei dati. Le **variabili**; il loro **nome** e il loro **valore**. L'area degli indirizzi delle variabili.

2. Dal problema al programma:

Il percorso per passare dal problema alla sua risoluzione da parte del calcolatore:

l'analisi del problema, l'individuazione del percorso, la codifica, l'istruzione del calcolatore, l'esecuzione

2.a - l'analisi del problema

L'**analisi del testo** (la lettura, l'individuazione del contesto e dello scopo, l'individuazione delle informazioni esplicite ed implicite). Il **grafo di scomposizione** del problema. L'**analisi dei dati**: i dati **costanti** e i dati **parametrici d'entrata, d'uscita e interni**. Nomi dei dati e nomi delle variabili.

2.b - il percorso / la codifica

L'individuazione del percorso (**processo**). L'**ingresso**, l'**elaborazione**, l'**uscita**. L'attività di codifica con il **linguaggio di progetto** e il **GWBASIC**. Le proposizioni d'ingresso (**chiedi / INPUT**), di elaborazione (**ottiene... facendo... / LET... =...**) e d'uscita (**scrivi / PRINT**). L'uso della scheda di programmazione.

2.c - l'istruzione del calcolatore / l'esecuzione

I **codici di macchina**. Il ruolo del numero di linea (**puntatore**). Il controllo del listato (**LIST**). L'**esecuzione** di prova con **RUN** e l'uso di **NEW** e **CLEAR**. La gestione dei disk drive con **CHDIR** e **FILES** e dei programmi con **SAVE**, **LOAD**, e **KILL**.

3. Dati alfanumerici:

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 allo schermo (la **memoria caratteri**, la **scheda grafica**). La modalità "testo" in MS-DOS.

3.a - dati alfanumerici nel programma

Le **situazioni comunicative** che si sviluppano nell'attività con il calcolatore. La chiara richiesta di dati e la chiara comunicazione (**LOCATE**). I colori con **COLOR** e la pulizia con **CLS**. La comunicazione per il programmatore con **REM**. In **VISUAL BASIC**

3.b - le variabili alfanumeriche o "stringa"

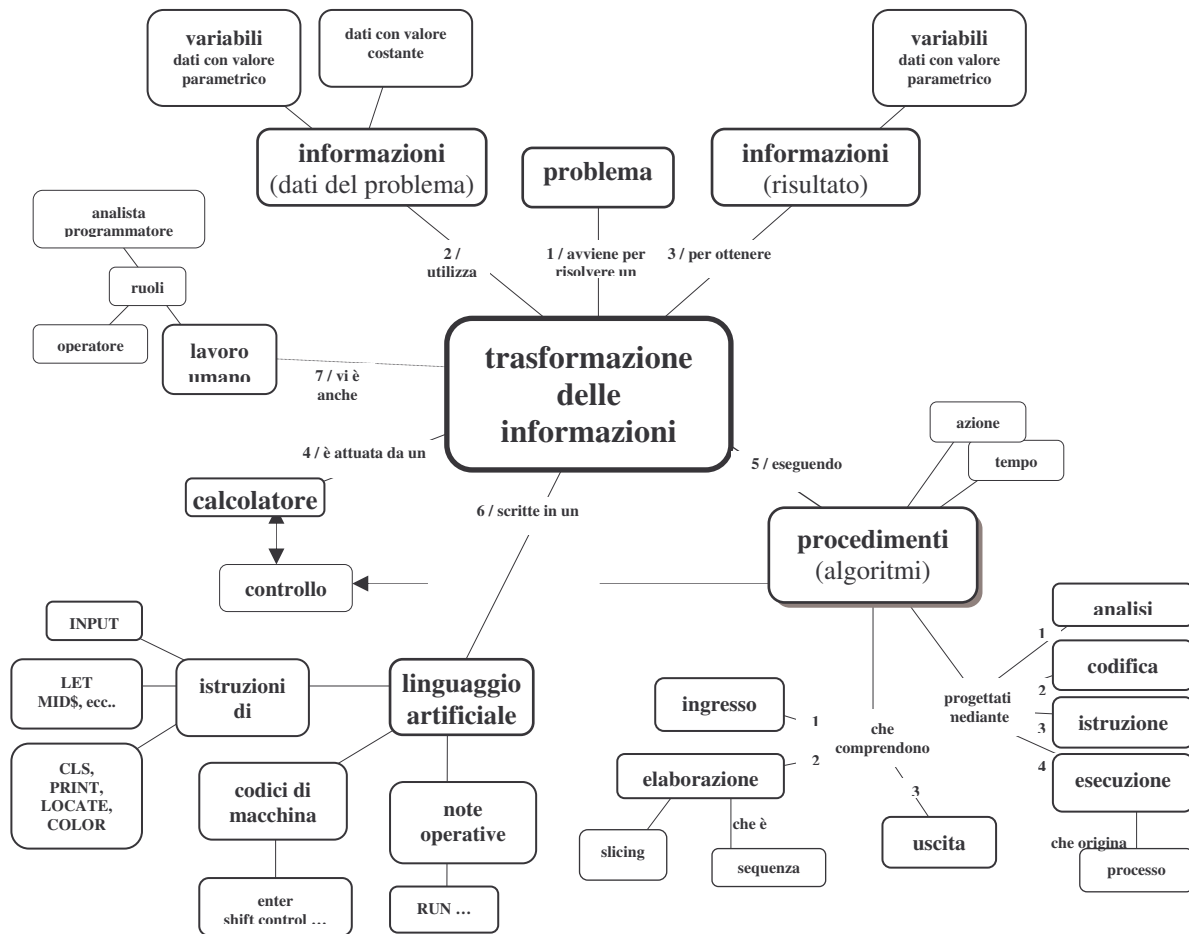
L'inserimento di testi dentro una variabile. La struttura delle variabili alfanumeriche. Lo **slicing** con **MID\$**, **LEFT\$** e **RIGHT\$**. La "misurazione" delle variabili alfanumeriche con **LEN**. Il passaggio da un tipo all'altro con **VAL** e **STR\$**. In **VISUAL BASIC**

3.c - 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#**), l'operazione di **chiusura** (**CLOSE**).

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

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



una trasformazione delle informazioni avviene ...

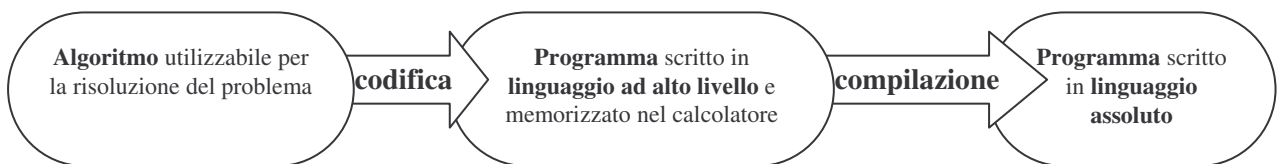
• **presentazione**

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 il quale, dopo averlo **compilato** in linguaggio in **linguaggio assoluto** potrà risolverlo ogni volta che sarà necessario.

In altre parole impareremo a **programmare un calcolatore** affrontando però problemi più semplici, quelli che solitamente siamo abituati ad affrontare nell'attività di matematica.

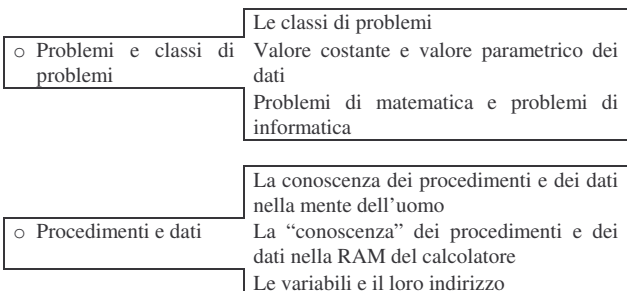
* dal nome del matematico *Al Khwarizmi*, vissuto a Baghdad dal 750 al 830(?) ed autore del libro "Kitab al jabr wa'l mugabala" nel quale erano raccolte importanti nozioni di matematica di derivazione babilonese, indù ed alessandrina. Dalla distorsione del suo nome sono derivati i termini algoritmo e 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 da un algoritmo (cioè da un procedimento studiato per risolvere un problema) un programma grazie al quale sarà il calcolatore a risolvere il problema.



fase 1 L'AMBIENTE DI LAVORO

- gli ambienti di lavoro
- il caricamento del sistema operativo
- il lavoro in diretta



- il lavoro umano

• **gli ambienti di lavoro (MS-DOS e WINDOWS)**

Per ora lavoreremo utilizzando il vecchio sistema operativo MS-DOS (diffuso negli anni 80') e il linguaggio GWBASIC uno dei più usati, in quegli anni, dai programmatori che realizzavano applicazioni per i calcolatori. Questo perché l'**ambiente di lavoro** e la grafica sono molto semplici e permettono di concentrarci sui problemi da risolvere. Proprio perché opera in un ambiente di lavoro più semplice, il linguaggio GWBASIC è limitato ed essenziale. Ben diverso sarà il VISUAL BASIC che deve gestire un ambiente complesso come quello di WINDOWS con grafica, suoni, immagini, filmati, ecc...

```

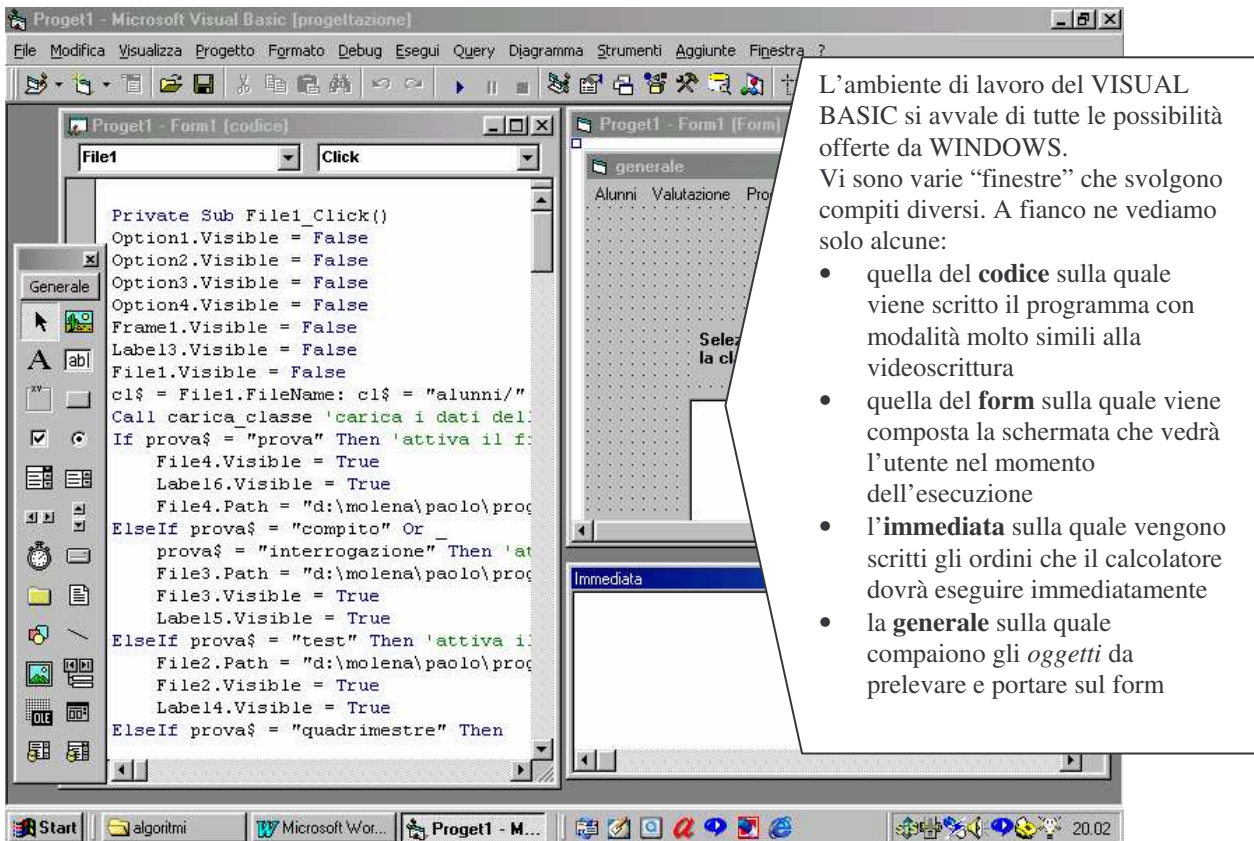
LIST
5 CLS
10 INPUT"Scrivi il prezzo di vendita al chilo":PVEND
20 INPUT"Scrivi il n° di chili acquistati":KACQ
30 INPUT"Scrivi il n° di chili scartati":KSCART
40 INPUT"Scrivi il prezzo di acquisto al chilo":PACQ
50 LET KVEND=KACQ-KSCART
60 LET RICVEND=PVEND*KVEND
70 LET SACQ=PACQ*KACQ
80 LET GUAD=RICVEND-SACQ
85 CLS
90 PRINT"Il guadagno del fruttivendolo è di euro ";GUAD
OK
    
```

L'ambiente di lavoro del GWBASIC risente delle limitate possibilità del sistema operativo MS-DOS.

Vi è un unico schermo sul quale saranno scritte:

- le istruzioni del programma (che vanno sempre precedute dal numero di linea) e che saranno memorizzate dal calcolatore in attesa dell'ordine di esecuzione
- gli ordini che il calcolatore dovrà eseguire subito.





• **il caricamento del s.o. MS-DOS e del linguaggio GWBASIC**

A questo punto possiamo inserire il dischetto consegnatoci dall'insegnante nel drive del calcolatore che poi potremo accendere. Questo dischetto è un **disco di sistema**, cioè un disco che, tra l'altro, contiene il sistema operativo MS-DOS. Il calcolatore una volta acceso inizia ad eseguire le istruzioni residenti nella ROM. In questa fase esso non è in grado di leggere i caratteri della tastiera e l'uso dello schermo è molto semplice. Dopo l'autodiagnosi arriva la lettura del dischetto e dunque il calcolatore legge da esso le istruzioni del sistema operativo e le colloca nella RAM rendendole subito operative. Sullo schermo compare:

A >

Con cui il calcolatore segnala di essere pronto a leggere altri dati dal dischetto (indicato convenzionalmente con A). Sarà ora necessario battere l'ordine:

➤ **GWBASIC** seguito dal tasto

Il calcolatore leggerà dal dischetto, collocherà nella RAM e renderà subito operative le istruzioni necessarie per poter scrivere programmi utilizzando questo linguaggio.

Con gli ordini:

➤ **KEY OFF** e seguito da **CLS** e

potrò togliere i disegni dei tasti funzione e poi pulire lo schermo.

• **il lavoro in diretta**

Tutti i linguaggi di programmazione danno anche la possibilità **lavorare in diretta**, di lavorare cioè in modo che il calcolatore esegua immediatamente il singolo ordine invece di memorizzarlo come parte di un programma. In GWBASIC basterà comporre il singolo ordine facendolo seguire dalla pressione di ; se l'istruzione non è preceduta da un numero (numero di linea) il calcolatore sa che l'ordine è da eseguire immediatamente. Questo consente anche di utilizzare il calcolatore come una normale calcolatrice facendo precedere i calcoli da effettuare dall'ordine PRINT (scrivi).

Se ad esempio batterò sulla tastiera:

➤ **PRINT 24 + 12** seguito da

sullo schermo apparirà il numero 36. 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 in BASIC 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:

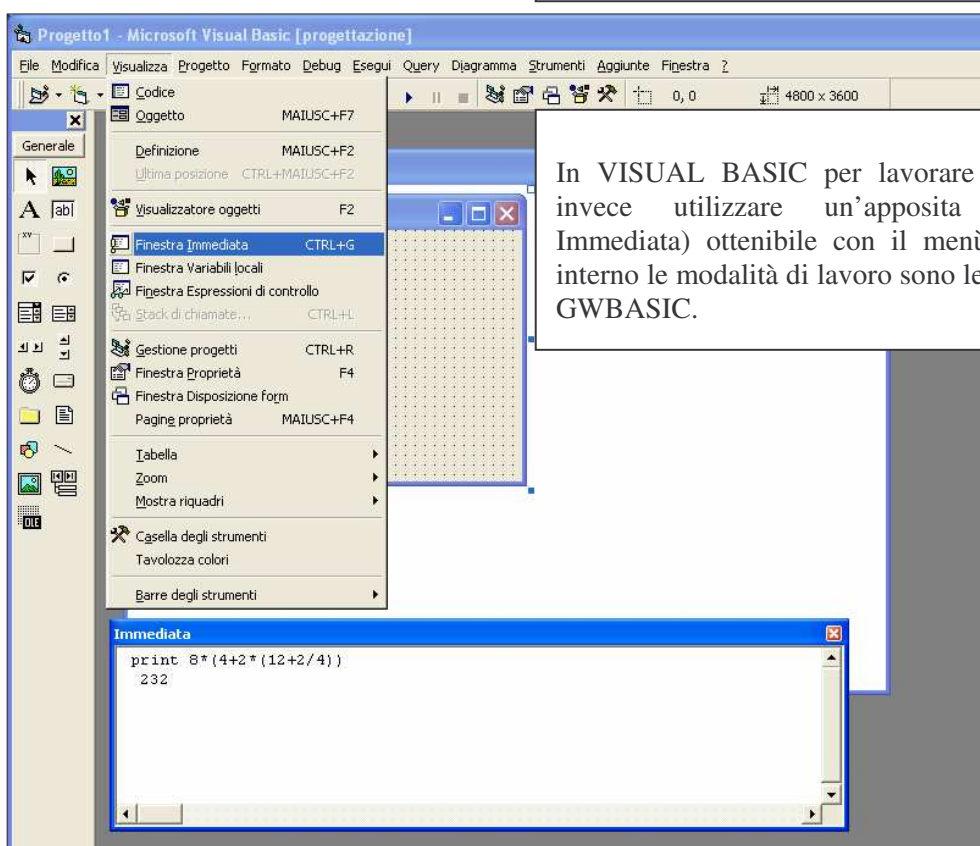
➤ **PRINT 8*(4+2*(12+2/4))** seguito da

sullo schermo apparirà 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**

potremo a questo punto esercitarci facendo risolvere al calcolatore alcune espressioni tratte dal libro di matematica.



In VISUAL BASIC per lavorare in diretta bisognerà invece utilizzare un'apposita finestra (Finestra Immediata) ottenibile con il menù Visualizza. Al suo interno le modalità di lavoro sono le stesse già viste per il GWBASIC.

Ma il calcolatore non è stato costruito per funzionare come calcolatrice ma per **essere programmato**. Cioè per essere istruito in modo da essere in grado di risolvere problemi.

Ma quando è conveniente far risolvere un problema da un calcolatore?

○ **Problemi e classi di problemi**

Le classi di problemi

Visto che, come vedremo, il lavoro di programmazione del calcolatore richiede tempo e fatica, realizzare un programma è conveniente solo quando esso potrà essere eseguito molte volte; ogni volta che si avrà a che fare con quel tipo di problemi. Un programma serve per risolvere una **classe di problemi**, cioè una serie di problemi che richiedono un comune procedimento di risoluzione anche se cambiano i valori dei dati. Per capire meglio proseguiamo il nostro lavoro utilizzando un semplice esempio.

Mettiamo di avere il seguente problema:

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?

Scrivi qua sotto il procedimento per risolverlo:

$100 * 1 = 100$

.....

.....

.....

Questo procedimento potrà anche essere tradotto in un linguaggio di programmazione e memorizzata in un calcolatore ma sarà poco conveniente. Perché?

.....

.....

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?

Riscrivi qua sotto il procedimento per risolverlo:

numero di chili di arance acquistate * prezzo di acquisto al chilo = spesa per l'acquisto delle arance

.....

.....

.....

Affidare la risoluzione di questo problema ad un calcolatore sarà più conveniente? Perché?

.....

.....

Valore costante e valore parametrico dei dati

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**.

<i>nome del dato</i>	<i>valore</i>
<i>n° di chili acquistati</i>	<i>100</i>
<i>n° di chili scartati</i>	<i>10</i>
<i>prezzo di acquisto al chilo in euro</i>	<i>1</i>
<i>prezzo di vendita al chilo in euro</i>	<i>2</i>

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>nome del dato</i>	<i>valore</i>
<i>n° di chili acquistati</i>	?
<i>n° di chili scartati</i>	?
<i>prezzo di acquisto al chilo in euro</i>	?
<i>prezzo di vendita al chilo in euro</i>	?

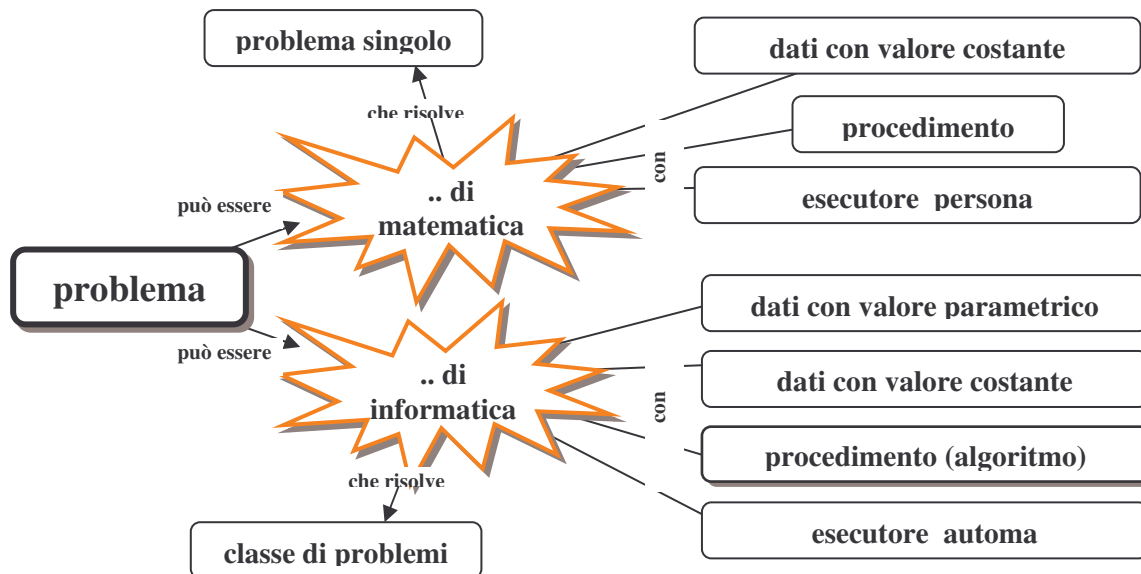
Problema di matematica e problema di informatica

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

il **problema di matematica**, cioè 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 (dichiarativi) con valore costante e dunque il fatto che vi sarà una sola esecuzione

il **problema di informatica**, che avrà come esecutore il calcolatore (spetta a lui trovare il risultato), dove il nostro compito è individuare la procedura di risoluzione e comunicarla al calcolatore. Caratteristica di questo tipo di problema è la prevalenza dei dati (dichiarativi) 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 il seguente schema riassuntivo:



Esercizio: distinguere nomi e valori di dati:

<i>nome del dato</i>	<i>valore del dato</i>	<i>nome del dato</i>	<i>valore del dato</i>
<i>colore dei capelli</i>	<i>rosso</i>		

ed ora al lavoro!

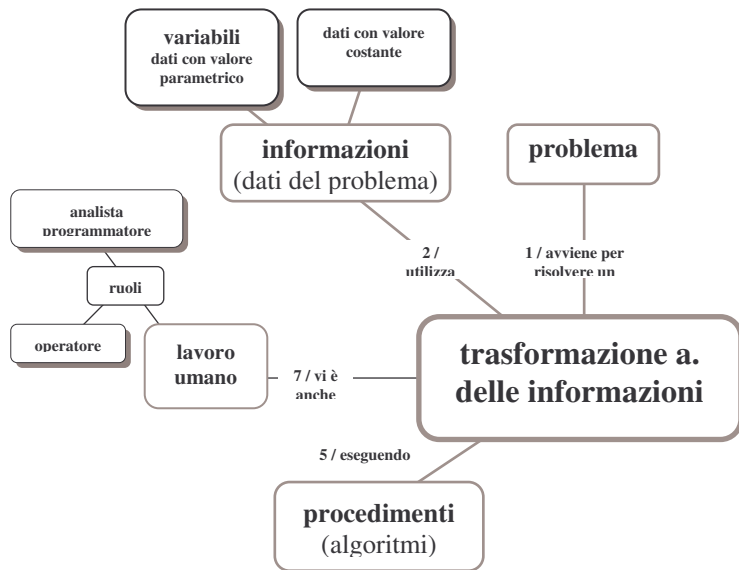
a) Esercizi: trasformare un problema di matematica in problema di informatica

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 se percorre abitualmente 100 km al giorno?

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?

b) Osserva la mappa a fianco, che evidenzia gli argomenti trattati nelle schede precedenti, e rispondi alle domande:

Il testo di un problema di informatica può ospitare valori di dati costanti? Quando? _____



○ **Procedimenti e dati**

La conoscenza dei procedimenti (procedurale) e quella dei dati (dichiarativa) 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 relativamente ai problemi di informatica anche in questi casi noi conosciamo bene 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 un procedimenti velocemente e senza pensarci sopra.

Si definisce invece con il termine di **conoscenza dei dati (conoscenza dichiarativa)** 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 di informatica, 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

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

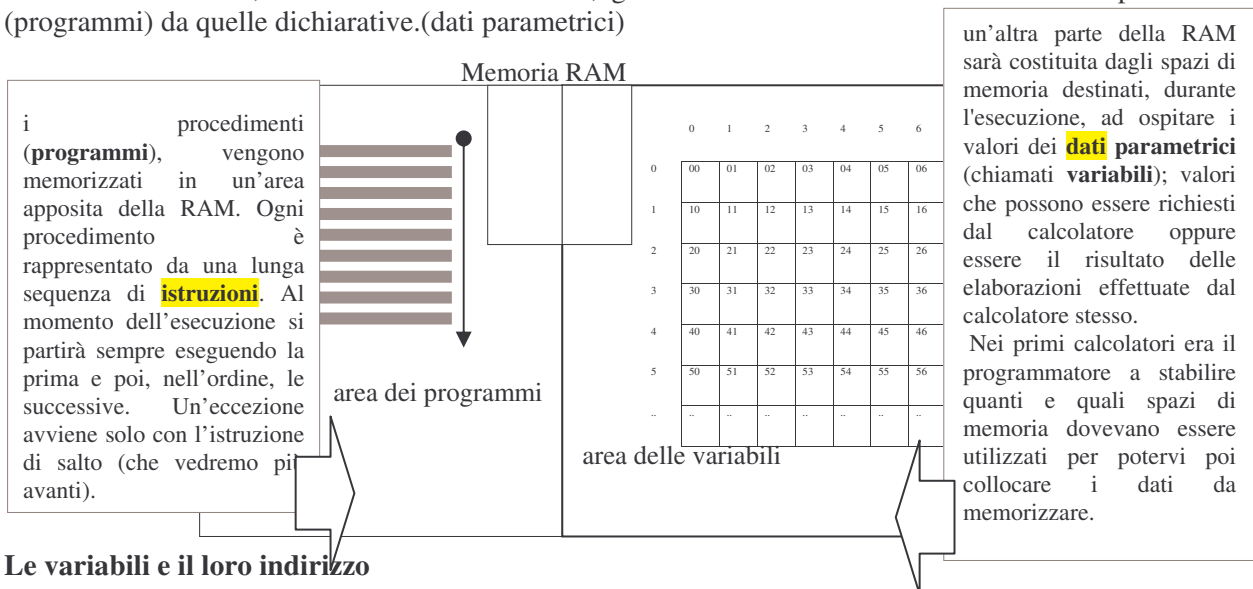
Mentre il procedimento è stabilmente memorizzato nella parte di memoria che ospita la conoscenza procedurale e potrà essere eseguito senza pensarci sopra, i valori dei dati, chiesti nella una fase iniziale, dovranno essere ospitati in quella parte di memoria in cui custodiamo la conoscenza dichiarativa.

La “conoscenza” dei procedimenti e quella dei dati nella RAM del calcolatore: i programmi, le variabili

Il calcolatore memorizza le informazioni in modo radicalmente diverso. Dobbiamo immaginare la memoria di un calcolatore come una cassetiera con tanti cassette 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.

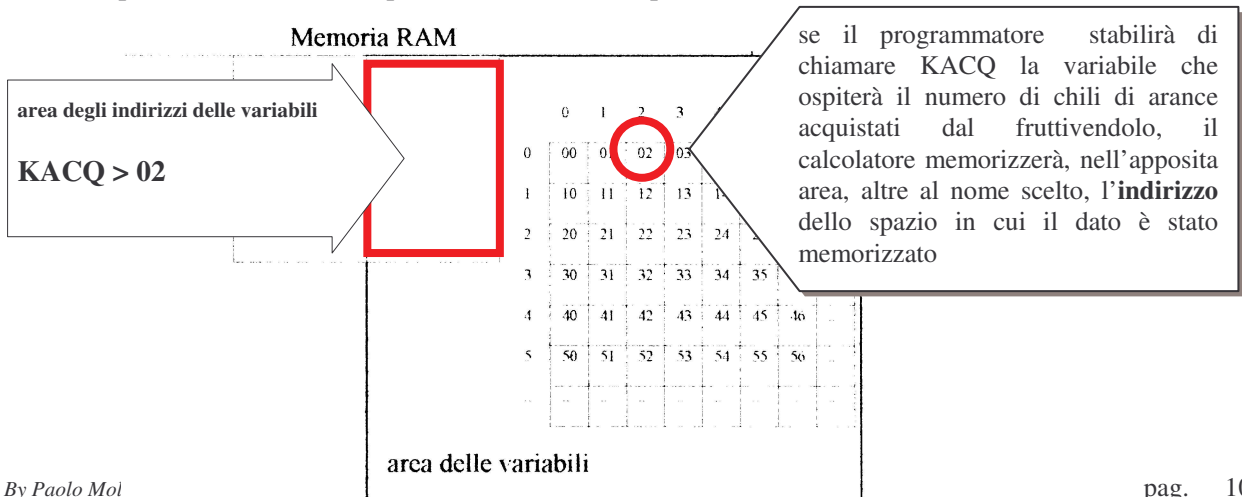
Anche il calcolatore, come la mente dell'uomo, gestisce in modo diverso le conoscenze procedurali (programmi) da quelle dichiarative.(dati parametrici)



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 una 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. Inoltre i nomi non possono corrispondere ad istruzioni del linguaggio che si sta usando. Per evitare confusioni al fianco puoi trovare un elenco completo degli ordini GWBASIC

In buona parte delle versioni di BASIC il nome seguito dal segno % (es. Perim%) indica una variabile destinata ad ospitare solo numeri interi (variabile **integer**). Questa variabile utilizza solo due byte della memoria RAM invece dei quattro utilizzati dalle normali variabili numeriche.

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 VISUAL BASIC).

Esempi di nomi validi:

perim34
PERIM_TOT
printi

non validi:

12aso
PERIM TOT
print

Elenco degli ordini GWBASIC

ABS, ASC, ATN, AUTO, BEEP, BLOAD, BSAVE, CALL, CDBL, CHAIN, CHDIR, CHR\$, CINT, CIRCLE, CLEAR, CLOSE, CLS, COLOR, COM, COMMON, CONT, COS, CSNG, CSRLIN, CVI, CVS, CVD, DATA, DATE\$, DEF FN, DEFINT, DEFSNG, DEFDBL, DEFSTR, DEF SEG, DEF USR, DELETE, DIM, DRAW, EDIT, END, ENVIRON, ENVIRON\$, EOF, ERASE, ERDEV, ERDEV\$, ERR, ERL, ERROR, EXP, EXTERR, FIELD, FILES, FIX, FOR NEXT, FRE, GET, GOSUB RETURN, GOTO, HEX\$, IF, INKEY\$, INP, INPUT, INPUT#, INPUT\$, INSTR, INT, IOCTL, IOCTL\$, KEY, KILL, LEFT\$, LEN, LET, LINE, LIST, LLIST, LOAD, LOC, LOCATE, LOCK, LOF, LOG, LPOS, LPRINT, LPRINT USING, LSET, RSET, MERGE, MID\$, MKDIR, MKI\$, MKS\$, MKD\$, NAME, NEW, OCT\$, ON, OPEN, OPTION BASE, OUT, PAINT, PALETTE, PALETTE, PCOPY, PEEK, PEN, PLAY, PMAP, POINT, POKE, POS, PRESET, PSET, PRINT, PRINT#, PUT, RANDOMIZE, READ, REM, RENUM, RESET, RESTORE, RESUME, RETURN, RIGHT\$, RMDIR, RND, RUN, SAVE, SCREEN, SGN, SHELL, SIN, SOUND, SPACE\$, SPC, SQR, STICK, STOP, STR\$, STRING, STRING\$, SWAP, SYSTEM, TAB, TAN, TIME\$, TIMER, TRON, TROFF, UNLOCK, USING, USR, VAL, VARPTR, VARPTR\$, VIEW, WAIT, WHILE, WEND, WIDTH, WINDOW, WRITE, WRITE#

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.

Nota Bene: in GWBASIC 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.

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:



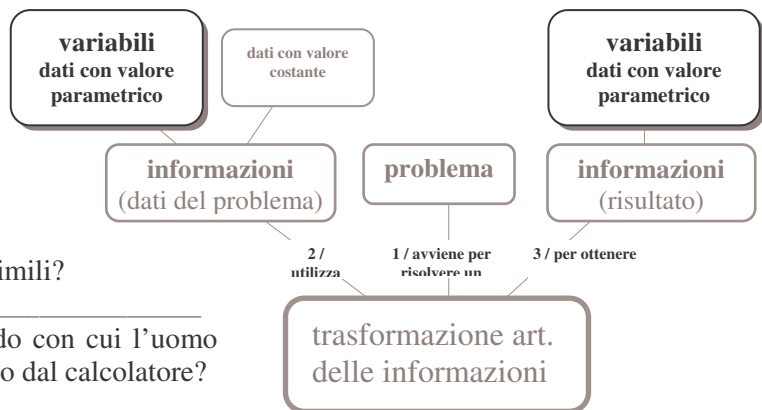
In sintesi ...

In quale memoria del calcolatore e in quale sua area sono ospitati i dati evidenziati nella mappa al fianco?

A quali tipi di conoscenza dell'uomo sono simili?

Quale 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?



DAL PROBLEMA AL PROGRAMMA

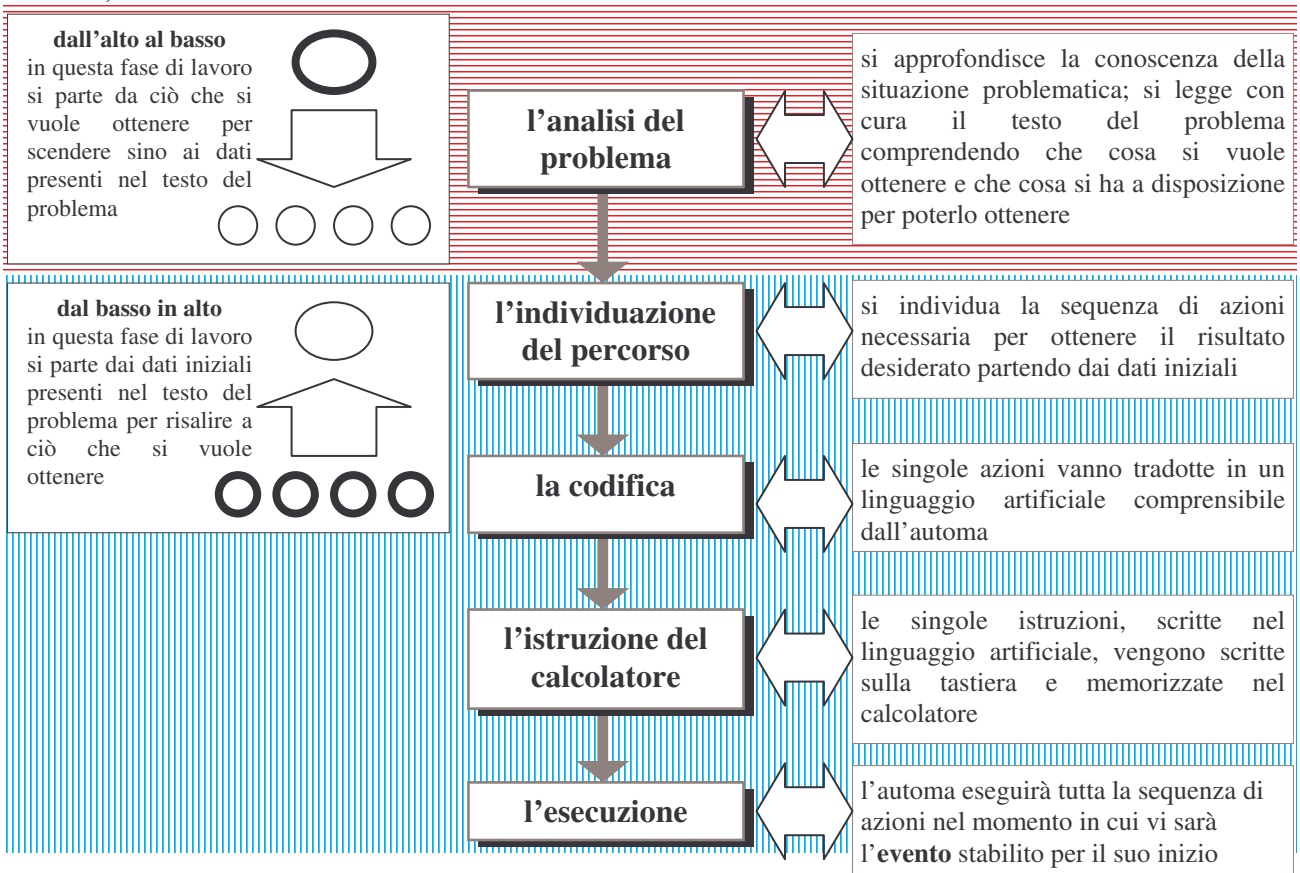
fase 2 DAL PROBLEMA AL PROGRAMMA

presentazione

Vediamo come arrivare da un problema ad una sequenza di istruzioni eseguibili dal calcolatore:

<ul style="list-style-type: none"> presentazione <ul style="list-style-type: none"> L'analisi del problema 	l'analisi del testo il grafo di scomposizione del problema l'analisi dei dati
<ul style="list-style-type: none"> <ul style="list-style-type: none"> L'individuazione del percorso utilizzando il <i>linguaggio di progetto</i> 	le proposizioni di ingresso le proposizioni di elaborazione le proposizioni di uscita
<ul style="list-style-type: none"> <ul style="list-style-type: none"> La codifica utilizzando il <i>Gwbasic</i> 	le proposizioni di ingresso con INPUT le proposizioni di elaborazione con LET le proposizioni di uscita con PRINT
<ul style="list-style-type: none"> l'uso della scheda di programmazione <ul style="list-style-type: none"> L'istruzione del calcolatore L'esecuzione 	l'istruzione e i codici di macchina l'esecuzione e le note operative

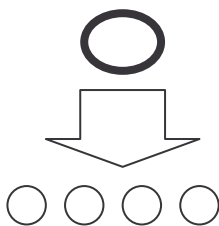
Individuiamo innanzitutto cinque attività: **l'analisi del problema, l'individuazione del percorso, la codifica, l'istruzione del calcolatore ed infine l'esecuzione.**



La mappa a fianco illustra il percorso necessario per progettare il procedimento da utilizzare nella risoluzione di un problema. Completa la verbalizzazione seguente:

La trasformazione artificiale delle informazioni avviene per risolvere un problema, utilizza delle informazioni (dati del problema) per ottenere _____





L'analisi del problema

Come abbiamo appena visto, in questa prima fase si approfondisce la situazione problematica. Nella realtà questo approfondimento è piuttosto complesso in quanto comporta il recupero e l'analisi di tutte le informazioni che, in qualche modo, possono influire sulla soluzione del problema.



se il problema è *costruire una casa*, l'attività di analisi consisterà in:

- studiare con cura le esigenze di chi utilizzerà la casa; sia le esigenze esplicite (cioè quelle che vengono comunicate al progettista) sia quelle presumibili (cioè quelle non comunicate in modo esplicito)
- studiare il regolamento edilizio del Comune e le indicazioni del Piano regolatore verificando cosa è possibile fare e in che modo e con quali costi di concessione
- approfondire le consuetudini edilizie del luogo, gli stili di costruzione più usati, i materiali più caratteristici
- analizzare la facilità di procurarsi sul posto i vari materiali e i relativi costi
- verificare la presenza sul posto di imprese di costruzione e di artigiani specializzati nella costruzione dei vari impianti; verificarne competenze e costi
- ecc...

A scuola la situazione problematica viene, in genere, comunicata con un testo (il testo del problema) che contiene già, oltre all'enunciazione del problema, tutti i dati che saranno necessari per individuare il procedimento di risoluzione.

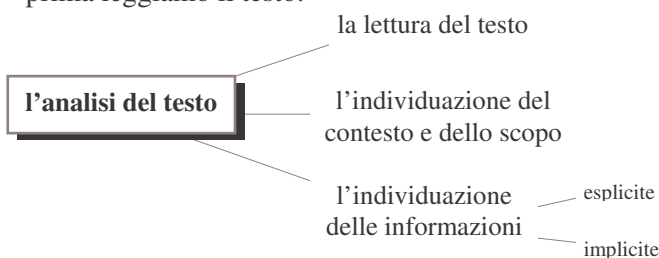
Questo può rendere più semplice l'attività di analisi ma anche la lettura di un testo può comportare delle difficoltà:

- il testo può essere scritto male, in modo poco comprensibile, utilizzando termini ambigui
- chi legge il testo può essere distratto o superficiale; una parte considerevole di errori nella risoluzione dei problemi deriva da una cattiva lettura del testo (il lettore ha in mente altri procedimenti di risoluzione e non riesce a concentrarsi su quanto gli viene chiesto)

L'analisi del testo

Iniziamo dunque dall'**analisi del testo**:

- prima leggiamo il testo:



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?

- cerchiamo poi di capire, e dunque di ripetere a noi stessi, a che cosa serve questo problema e di che cosa si parla. E' bene ricordare che sarà impossibile insegnare ad altri (il calcolatore) come risolvere un problema se non lo capiamo noi.
- sottolineiamo poi nel testo del problema tutti i dati di tipo dichiarativo

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?

- cerchiamo di capire inoltre *ciò che è indispensabile capire per individuare il procedimento di risoluzione*

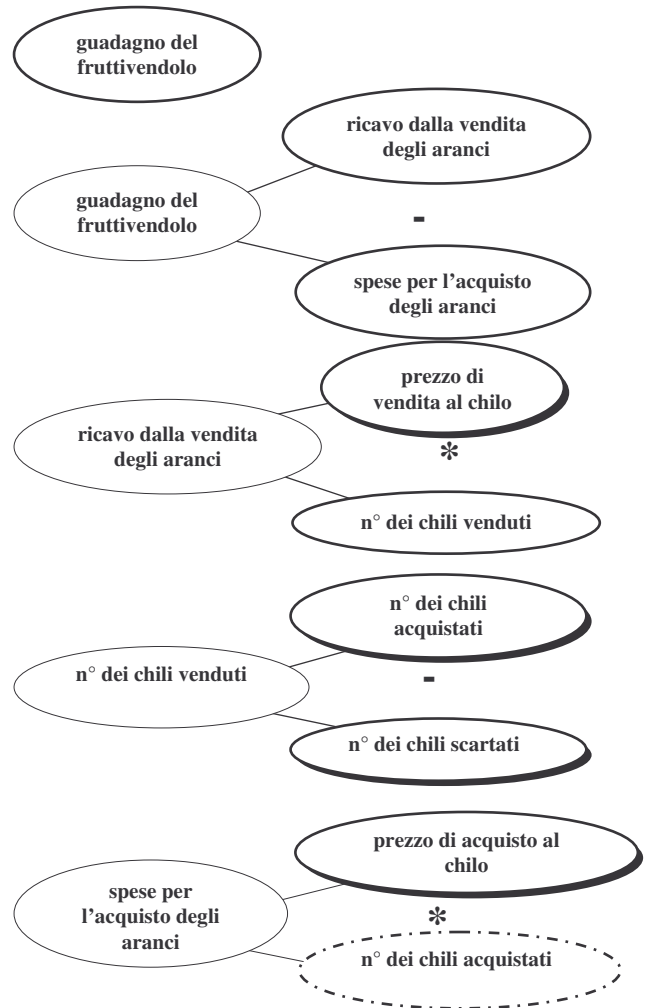
e che il testo del problema non dice e che comunque non è ricavabile dal calcolatore durante l'esecuzione. Dovrà conoscere dunque il n° di chili di arance acquistate, il n° di chili di arance scartate, il prezzo di acquisto al chilo e il prezzo di vendita al chilo. Chiamo **dati parametrici d'entrata** quei dati il cui valore deve essere chiesto dall'esecutore prima di iniziare l'elaborazione.

- Il n° di chili di arance vendute non va chiesto in quanto è ricavabile dal calcolatore durante l'esecuzione visto che si conoscono il n° di chili di arance acquistate e il n° di chili di arance scartate.

Il grafo di scomposizione del problema

Passiamo poi a realizzare il **grafo di scomposizione del problema** con cui viene effettuata la scomposizione *top/down* del problema; partendo cioè da ciò che si intende ottenere. Esso viene realizzato con i seguenti passaggi:

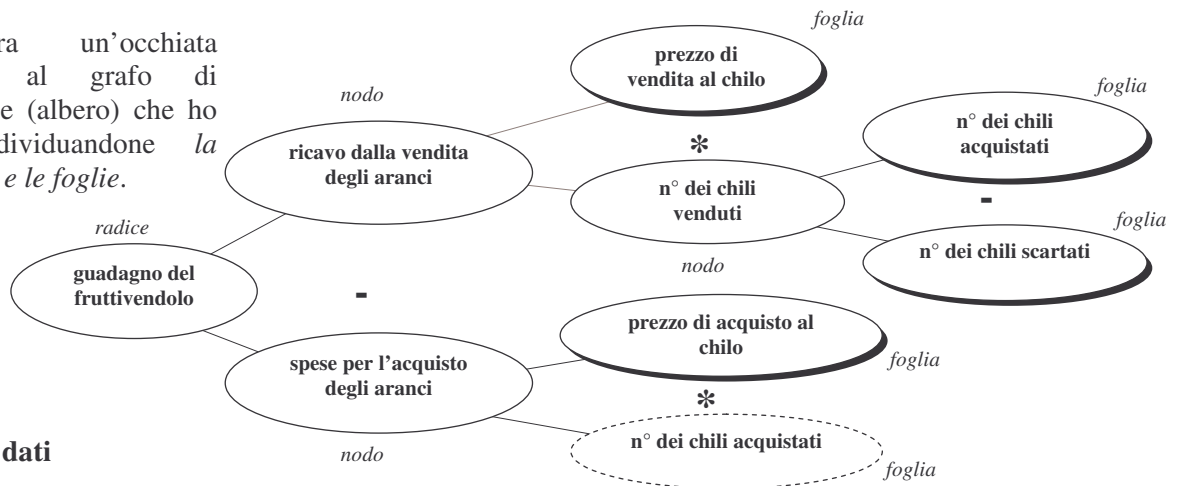
- individuazione del prodotto (*radice* del grafo) detto anche dato parametrico d'uscita
- individuazione del primo livello di scomposizione (*nodi* del grafo); come spesso succede in problemi di questo tipo, il guadagno si ottiene facendo *ricavi* meno *costi*
- affrontiamo la scomposizione del primo dei due nodi (in questo caso è indifferente partire dall'uno o dall'altro)
- dei due dati individuati uno (*prezzo di vendita al chilo*) è un dato iniziale (dato parametrico d'entrata o che sono presenti nel testo del problema / *foglia* nel grafo). L'altro dato (*n° dei chili venduti*) è, come abbiamo già visto, un nodo scomponibile in due dati iniziali.
- completata la scomposizione del ramo collegato a *ricavo dalla vendita degli aranci*, potremo passare a scomporre l'altro ramo



il segno tratteggiato utilizzato per il *n° dei chili acquistati* serve a ricordarci che questo dato parametrico d'entrata è già presente nel grafo.

- avendo ottenuto subito due dati iniziali, il lavoro di scomposizione è terminato

Diamo ora un'occhiata complessiva al grafo di scomposizione (albero) che ho ricavato individuandone la radice, i nodi e le foglie.



L'analisi dei dati

Ed infine si procede all'analisi dei dati che viene fatta utilizzando il grafo di scomposizione. Tutti i dati vengono classificati in:

- dati costanti** **dc**
- dati parametrici d'entrata** (richiesti dal calcolatore all'inizio dell'elaborazione) **dpe**
che nell'albero di scomposizione corrispondono alle foglie
- „ **d'uscita** (corrispondono al prodotto) **dpu**
che corrispondono alla radice
- „ **interni** (altri dati risultati dalle elaborazioni) **dpi**
che corrispondono ai nodi

Abbiamo così completato la fase di analisi del problema. Il grafo di scomposizione, così completato, sarà fondamentale per individuarne il procedimento di risoluzione (in questo problema non sono presenti dati costanti).



I nomi dei dati vanno riportati in una tabella da collocare sotto il grafo.

nome del dato	tipo di dato	nome della variabile
prezzo di vendita al chilo	parametrico d'entrata	PVEND
n° dei chili acquistati	parametrico d'entrata	KACQ
n° dei chili scartati	parametrico d'entrata	KSCART
prezzo di acquisto al chilo	parametrico d'entrata	PACQ
n° dei chili venduti	parametrico interno	KVEND
ricavo dalla vendita delle arance	parametrico interno	RICVEND
spese per l'acquisto delle arance	parametrico interno	SPEACQ
guadagno del fruttivendolo	parametrico d'uscita	GUAD

il tratteggio serve a segnalare che questo dato è già stato acquisito e

Prima della stesura del programma verranno inseriti in questo spazio i nomi delle variabili che saranno utilizzate.
E' bene scrivere tutto in matita in modo da poter effettuare correzioni.

nomi dei dati e nomi delle variabili

Nel lavoro di analisi del problema e di analisi dei dati va posta particolare attenzione al **nome dei dati** che deve essere *preciso e completo* in modo da evitare ogni ambiguità e possibile confusione con altri dati presenti nel problema.

Nella nostra mente ogni dato è collegato a tutti quei dati che possono avere con esso un legame di significato; un nome preciso ci permette di definire al meglio il dato evitando errori.



Le variabili sono invece dei semplici contenitori il cui nome è scelto dal programmatore. E' importante che sia breve e potrà richiamarci al dato in essa contenuto.

Una variabile può anche essere **riassegnata**, cioè possono essere depositati in successione dati diversi ognuno dei quali cancella il dato depositato in precedenza.



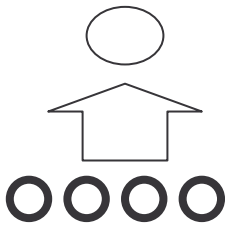
In sintesi ...

Quale è l'attività di cui ci siamo occupati in questa attività?

Nella realtà del mondo del lavoro chi si occupa di svolgere questa il lavoro di analisi?

Quale tipo di capacità deve possedere chi intende svolgere questo lavoro?



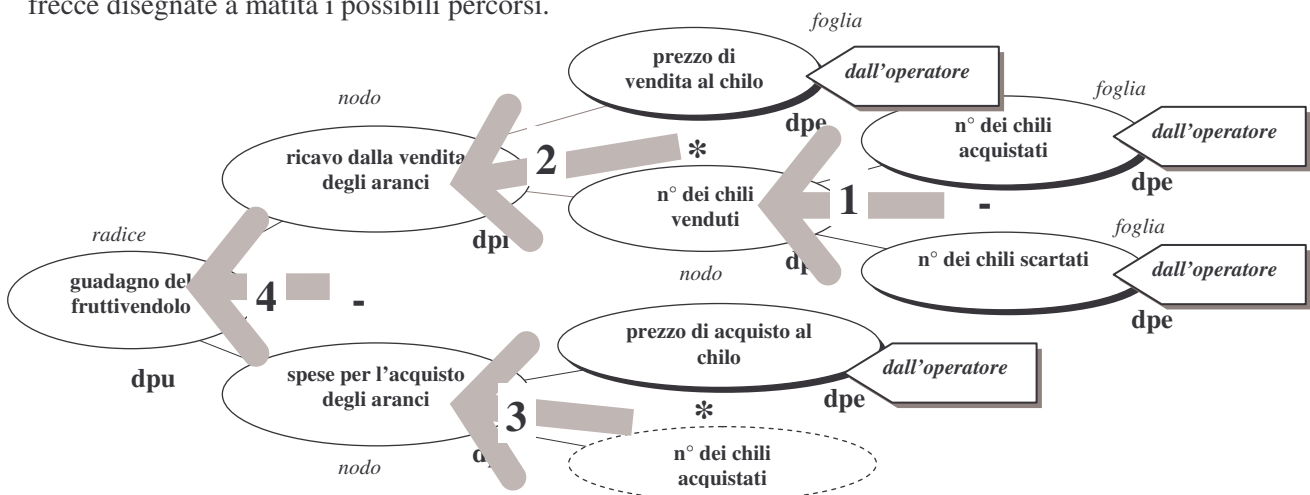


L'individuazione del percorso * (utilizzando il linguaggio di progetto)

Una volta analizzato il problema bisogna individuarne un procedimento di risoluzione; individuare cioè una sequenza di azioni che permetta all'esecutore, partendo dai dati iniziali, di arrivare al prodotto (parametro d'uscita). Questo lavoro verrà svolto in un italiano semplificato che chiameremo **linguaggio di progetto**, Solo successivamente, quando avremo ben chiaro ciò che intendiamo comunicare al calcolatore, passare alla codifica in basic.

* la sequenza di azioni che viene effettivamente svolta per andare dall'ingresso dei dati sino alla risoluzione del problema viene anche chiamata **processo**

La parte iniziale di questa attività potrà essere svolta direttamente sul grafo di analisi evidenziando con delle frecce disegnate a matita i possibili percorsi.



I numeri segnati sulle frecce indicano l'ordine con cui le azioni andranno svolte. Esse andranno a formare le **proposizioni (*) di elaborazione**. In esse l'istruzione di elaborazione sarà completata dall'indicazione dei dati da elaborare.

* in educazione linguistica il termine **proposizione** è sinonimo di frase

ad esempio: ottieni il n° dei chili venduti facendo n° dei chili acquistati meno n° dei chili scartati

Queste saranno precedute dalle proposizioni con cui il calcolatore chiederà all'operatore (cioè a colui che è seduto davanti alla tastiera al momento dell'elaborazione) i valori dei dati parametrici e li inserirà all'interno di variabili (**ingresso dei dati**) e seguite da quella (o quelle) con cui comunicherà allo stesso operatore il risultato dell'elaborazione (**uscita di dati**). Vediamo ora con attenzione questi tre tipi di proposizioni applicandole al problema di cui ci stiamo occupando.



Le proposizioni di ingresso

dato il cui valore deve essere conosciuto dall'esecutore.

Dobbiamo prevedere, all'inizio dell'esecuzione l'acquisizione, da parte dell'esecutore, dei valori dei dati parametrici d'entrata (d.p.e.). L'**ingresso** del valore di un dato deve, ovviamente, precedere la sua elaborazione. Per l'ingresso di informazioni usiamo il verbo **chiedi** seguito dal nome del

- chiedi il prezzo di vendita al chilo*
- chiedi il numero di chili acquistati*
- chiedi il numero di chili scartati*
- chiedi il prezzo di acquisto al chilo*

Le proposizioni di elaborazione

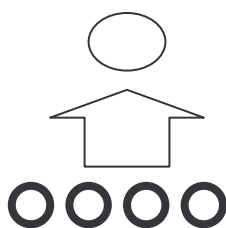
Come abbiamo visto nell'esempio per l'istruzione di elaborazione si utilizzerà il verbo **ottiene** seguito dal nome del dato che si vuole ottenere. Il verbo **facendo** sarà seguito dall'elaborazione che si intende realizzare.

ottiene il n° dei chili venduti facendo il n° dei chili acquistati meno il n° dei chili scartati
ottiene il ricavo dalla vendita degli aranci facendo prezzo di vendita al chilo per n° dei chili venduti
ottiene le spese per l'acquisto delle arance facendo prezzo di acquisto al chilo per n° dei chili acquistati
ottiene il guadagno del fruttivendolo facendo il ricavo dalla vendita delle arance meno le spese per l'acquisto delle arance

Le proposizioni di uscita

Alla fine dell'elaborazione il calcolatore ha ottenuto il risultato. Però bisogna ricordare che scopo dell'esecuzione non è il calcolo del risultato ma la sua comunicazione all'operatore. Il prodotto finale dell'elaborazione (o i prodotti nel caso che vi siano più dati richiesti) vengono comunicati utilizzando il verbo **scrivi** seguito dal nome del dato.

scrivi il guadagno mensile del fruttivendolo



La codifica (utilizzando il Gwbasic)

Quando saremo sicuri della procedura in linguaggio di progetto potremo occuparci della sua traduzione (**codifica**) in un linguaggio artificiale di alto livello (noi utilizzeremo il gwbasic). La codifica avverrà utilizzando gli appositi spazi della scheda che ci darà l'insegnante utilizzando gli ordini che il gwbasic ci mette a disposizione.

Le proposizioni di ingresso con INPUT

Per l'ingresso di informazioni **chiedi** sarà tradotto con l'ordine **INPUT** seguito dal nome della variabile in cui il valore sarà memorizzato. *Al momento dell'esecuzione il calcolatore farà comparire sul monitor un punto interrogativo e attenderà che l'operatore scriva il valore del dato seguito da invio (come si può notare ogni proposizione è preceduta da un numero di cui vedremo la funzione in seguito).*

10	INPUT PVEND
20	INPUT KACQ
30	INPUT KSCART
40	INPUT PACQ

Le proposizioni di elaborazione con LET

L'istruzione di elaborazione **ottiene** verrà tradotta con l'ordine **LET** seguito dalla variabile in cui sarà memorizzato il risultato dell'elaborazione. Il verbo **facendo** sarà sostituito dal segno = seguito dall'elaborazione che si intende realizzare (bisogna utilizzare i nomi delle variabili in cui sono memorizzati i dati).

50	LET KVEND = KACQ - KSCART
60	LET RICVEND = PVEND * KVEND
70	LET SPEACQ = PACQ * KACQ
80	LET GUAD = RICVEND - SPEACQ

Le proposizioni di uscita con PRINT

Il verbo **scrivi** viene tradotto con l'ordine PRINT seguito dal nome della variabile di cui si vuole scrivere sullo schermo il valore. La scrittura avverrà a partire dalla posizione di scrittura (indicata dal cursore) lasciando uno spazio bianco sia prima che, eventualmente, dopo il numero da scrivere; sarà seguita dall'invio della posizione di scrittura all'inizio della riga successiva.

90 PRINT GUAD

i segni di punteggiatura nell'uscita di dati

I segni di punteggiatura possono essere utilizzati per impedire l'invio a capo della posizione di scrittura dopo l'esecuzione dell'ordine PRINT. Con il punto e virgola la posizione di scrittura viene spostata subito dopo ciò che è stato eventualmente scritto. Con la virgola la posizione di scrittura viene spostata alla prima posizione di tabulazione disponibile.

Se ad esempio: A=10 B=20 C=30

Eseguendo in diretta PRINT A;B;C;

il calcolatore scriverà in successione i tre numeri e posizionerà la posizione di scrittura dopo il terzo

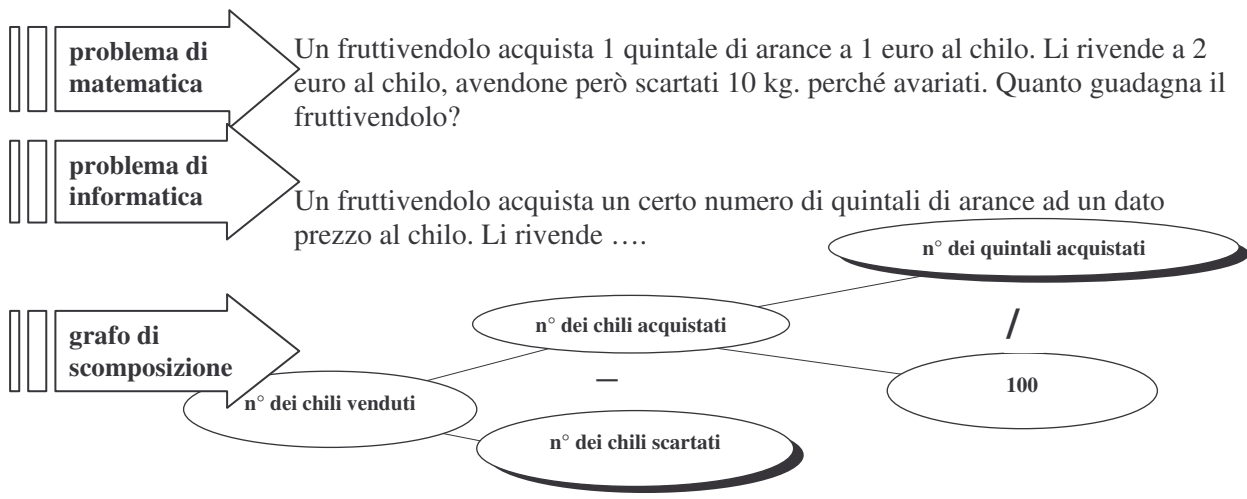
Eseguendo invece in diretta PRINT A,B;C

il calcolatore scriverà in successione i tre numeri e posizionerà la posizione di scrittura all'inizio della riga successiva

NB ricordare che dopo aver eseguito positivamente un ordine in diretta o un programma il calcolatore va comunque a capo e scrive OK

il cambio di unità di misura

Nei testi dei problemi di matematica è spesso presente la necessità di cambiare le unità di misura. Questo fatto comporta un'operazione aggiuntiva che dovrà comparire sia nel grafo di analisi che nel programma. Vediamo un esempio:



• l'uso della scheda di programmazione

Vediamo ora come utilizzare la scheda di programmazione, che sarà, d'ora in poi, lo strumento a nostra disposizione per svolgere il percorso che va dal problema al programma pronto per il calcolatore. Lo facciamo prendendo come esempio il problema che abbiamo appena analizzato:

1) Qui va scritto il testo del problema di matematica. Su questo testo vanno individuati quei dati costanti che è opportuno rendere parametrici.

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

2) Qui va scritto il testo del problema di informatica. Su di esso va svolta l'attività di analisi del testo.

Problema di informatica
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?

grafo di scomposizione e

3) Qui va disegnato il grafo di scomposizione del problema, partendo dal prodotto (ciò che si vuole ottenere). In questo spazio andrà anche svolta l'attività di analisi dei dati

analisi dei dati

4) In questa tabella andranno trascritti i nomi dei dati con, a fianco, la loro classificazione. I nomi delle variabili saranno decisi dopo la stesura della procedura in linguaggio di progetto e prima della stesura del programma.

procedura in linguaggio di progetto

programma

5) E' il momento di scrivere il procedimento di risoluzione. Lo si fa prima in linguaggio di progetto. Si parte osservando la destra del grafo di scomposizione: ad ogni foglia corrisponderà un ingresso di dati (leggi ...), mentre poi, procedendo da destra verso sinistra, i rami del grafo diventeranno proposizioni di elaborazione (ottiene ...). Ottenuto il prodotto finale non bisognerà dimenticare di scriverlo (scrivi ...).

6) Infine andrà scritto il programma. Ogni proposizione va collocata su una linea e deve essere preceduta da un numero di linea progressivo che andrà scritto prima del segno verticale più marcato. A partire da questo andrà scritta l'istruzione seguita dai dati

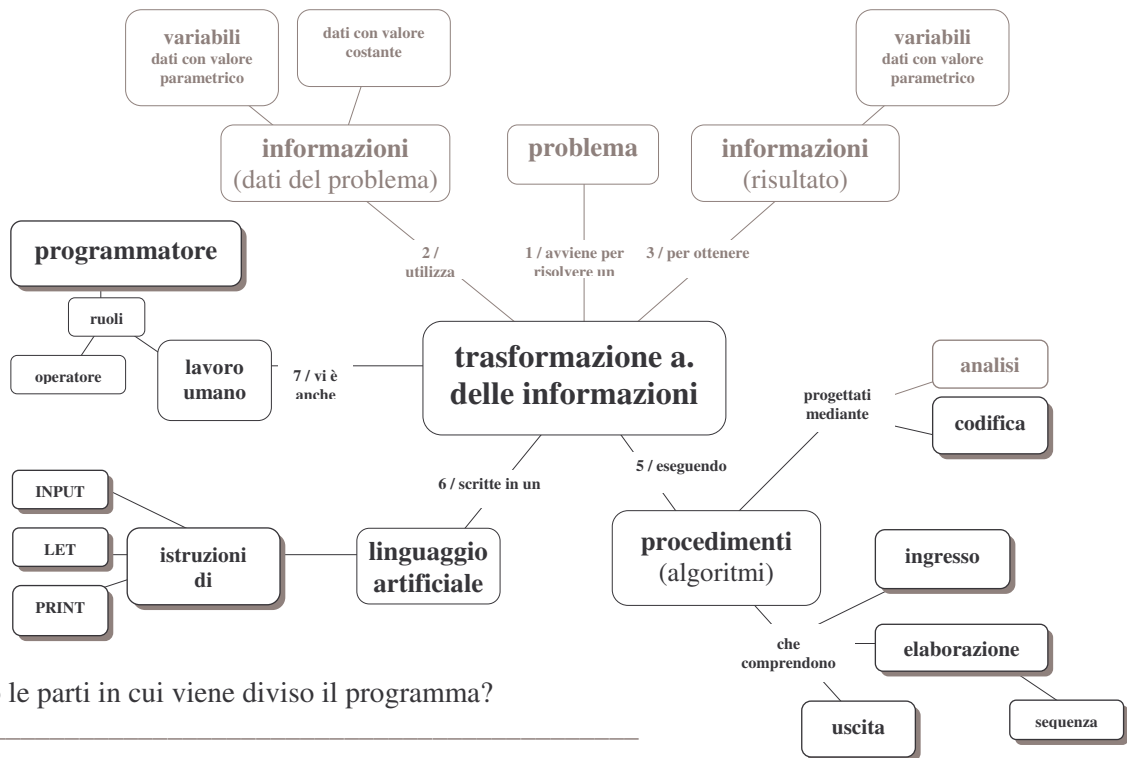
ed ora al lavoro !

Ora, utilizzando la scheda che ti ha consegnato l’insegnante, trascrivi il seguente problema di matematica e, facendo con cura tutti i passaggi, arriva al testo del programma.

*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**.*

in sintesi ...

La seguente mappa ci ricorda che in questa fase ci siamo occupati della parte centrale dell’attività di programmazione: quella in cui il programmatore, dopo l’analisi, individua il procedimento e lo codifica in un programma.



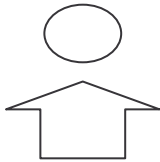
Quali sono le parti in cui viene diviso il programma?

Quale è il compito dell’ingresso di dati? Con quale ordine viene gestito? Come funziona quest’ordine?

Quale è il compito dell’elaborazione di dati? Con quale ordine viene gestita? Come funziona quest’ordine?

Quale è il compito dell’uscita di dati? Con quale ordine viene gestita? Come funziona quest’ordine?

Ti viene dato un problema di matematica. Quali sono i passaggi per arrivare all’esecuzione del programma?





L'istruzione del calcolatore

L'istruzione del calcolatore e i codici di macchina



Come prima cosa dovremo togliere dal drive il dischetto di sistema ed inserire quello di gruppo, sul quale d'ora in poi salveremo i programmi da noi realizzati.

Dovremo ora iniziare a trasferire il programma scritto su carta nella memoria del calcolatore in modo che esso possa essere eseguito tutte le volte che sarà necessario.

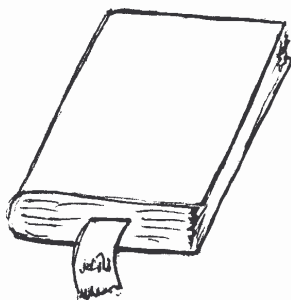
Questa attività viene chiamata **attività di istruzione** in quanto il suo scopo è quello di istruire il calcolatore su ciò che dovrà fare. Essa viene svolta tramite la tastiera, utilizzando i **codici di macchina** (così vengono chiamati i tasti dei caratteri e tutti i tasti speciali CTRL, , , Invio, Canc, ecc...).

Le modalità di istruzione sono spesso molto diverse a seconda del sistema operativo e della versione di BASIC utilizzati.

In GWBASIC:

- 1) vi è un unico schermo, ed ogni linea di programma (che normalmente contiene un'istruzione) viene scritta su di esso tramite la tastiera. Un rettangolino lampeggiante (cursore) indica la posizione nella quale si collocherà ciò che viene scritto.
- 2) il numero, collocato all'inizio di ogni linea, indica la priorità nell'esecuzione delle istruzioni del programma. Normalmente si numera di dieci in dieci in modo da poter inserire nuove linee tra quelle già battute.
- 3) una volta scritta sul video la linea dovrà essere inviata nella memoria del calcolatore con il tasto Invio. Se non si preme questo tasto la linea appare sullo schermo ma non è presente in memoria. Se viene cancellato lo schermo battendo l'ordine **CLS** (seguito da Invio), di ciò che è stato precedentemente scritto non rimarrà traccia.
- 4) ogni linea viene collocata in memoria tenendo conto del suo numero di linea e non dell'ordine con cui viene battuta. Ogni volta che una nuova linea viene inviata in memoria il calcolatore riorganizza le linee di programma collocandole in ordine crescente. Per questo motivo il numero di linea è obbligatorio e permette al calcolatore di distinguere la linea di programma, che non va eseguita subito ma memorizzata, dall'istruzione in "diretta" che viene eseguita subito e poi "dimenticata" dal calcolatore.
- 5) nel momento dell'invio in memoria la linea viene controllata ed eventualmente modificata (ad esempio tutti le istruzioni e i nomi delle variabili vengono registrati con i caratteri maiuscoli).
- 6) per cancellare una linea già in memoria bisognerà battere quel numero di linea seguito da Invio. Per modificarne il contenuto bisogna, una volta richiamata sullo schermo, spostare il cursore per riscrivere le parti da modificare, memorizzando poi la nuova versione con Invio.
- 7) su una linea si possono collocare più proposizioni; queste andranno però separate con il segno : (due punti) *esempio: 10 LET Kvend = Kacq - Kscart : PRINT Kvend*

il numero di linea come "puntatore"



Quando dobbiamo svolgere qualche ricerca per la scuola o per nostro interesse personale, ci può capitare di consultare enciclopedie e riviste. Se troviamo testi o immagini che ci possono interessare il modo più semplice per ritrovare ciò che ci interessa è porre quello che in informatica si chiama "puntatore". Inseriamo un pezzo di carta sul quale ci sarà un numero o una parola che hanno lo scopo di permetterci di raggiungere rapidamente le informazioni che ci interessano senza dover sfogliare tutto da capo.

Il numero di linea nei programmi BASIC, oltre che indicare al calcolatore l'ordine in cui verranno eseguite le diverse proposizioni, ha anche la funzione di puntatore. E' utile cioè per indicare al calcolatore la posizione da raggiungere in caso di spostamenti non sequenziali (*salti*). Verranno studiati nell'U.di A. PROGRAMMI).

il controllo del listato

Durante l'attività di istruzione o al termine di essa può essere necessario controllare ciò che è stato memorizzato, battendo in diretta l'istruzione:

➤ **LIST**

il calcolatore fornirà sullo schermo una copia precisa del programma in memoria.

il salvataggio del programma su disco

Se si vuole conservare il programma realizzato bisognerà registrarlo su disco, visto che tutte le informazioni presenti nella memoria RAM vengono cancellate allo spegnimento del calcolatore.

Battendo l'ordine:

nome con cui si vuole registrare il programma
➤ **SAVE" "**

si potrà registrare il programma presente nella RAM del calcolatore sulla memoria secondaria che in quel momento è operativa. E' bene dare ai programmi nomi che ne ricordino il contenuto.

Bisogna inoltre ricordare che, in MS-DOS, il nome non può superare gli otto caratteri più l'eventuale suffisso e che al suo interno non ci devono essere spazi bianchi.

Per salvare i due programmi studiati nella fase precedente potremo, ad esempio, usare:

➤ SAVE "V_FRUTTA" e
➤ SAVE "V_STOFFA"

- il salvataggio in formato testo

E' possibile salvare il programma in formato ASCII (cioè come testo) con l'ordine **SAVE"_" ,a**

Questo potrà essere utile se si vuole inserire il programma in un testo, oppure leggerlo in VISUAL BASIC. I file ASCII sono leggibili da tutti i programmi di videoscrittura.

- l'individuazione di percorsi

E' possibile leggere o salvare un programma su una memoria secondaria diversa da quello operativo facendo precedere il nome del programma dal nome del drive.

*Es. con **SAVE"a:Dati"** il calcolatore salverà il programma Dati sul floppy disk (chiamato a) anche se esso non è operativo.*

Quando la memoria secondaria è divisa in più cartelle (come accade sempre per l'hard disk) è possibile salvare all'interno di una particolare cartella indicando il percorso necessario.

Se, ad esempio; si vuole salvare il programma *Dati* sull'hard disk all'interno della cartella *IH* che a sua volta è contenuta nella cartella *lavori* si scriverà l'ordine: **SAVE"c:\lavori\IH\Dati**

il caricamento

Con l'ordine:

nome del programma che si vuole ottenere
➤ **LOAD" "**

il calcolatore leggerà dal disco le istruzioni del programma desiderato e le collocherà nella memoria RAM del calcolatore. Qualsiasi programma preesistente verrà cancellato e sostituito dal nuovo.

Con l'ordine:

➤ **FILES**

otterrò sullo schermo l'elenco di ciò che è registrato sul disco che in quel momento è operativo.

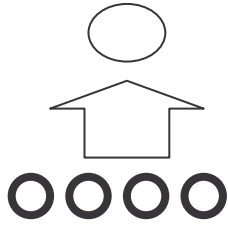
Per cancellare un programma dal disco si userà l'ordine (valido anche in VISUAL BASIC):

nome del programma che si vuole cancellare
➤ **KILL"_"**

il ritorno al sistema operativo

Se si vuole, al termine dei lavori, tornare al sistema operativo MS-DOS si dovrà battere l'ordine: **SYSTEM.**

E' importante ricordare di battere sempre SYSTEM prima di chiudere una finestra di GWBASIC aperta da WINDOWS



L'esecuzione

L'esecuzione del programma e le note operative

Terminata l'attività di istruzione il calcolatore è un automa programmato; prima di poter dire che il programma funziona dobbiamo provarlo. Alla richiesta del calcolatore, daremo dei dati semplici, in modo da poter verificare facilmente con carta e matita ciò che il calcolatore dà come risultato.

In *fase di esecuzione* potremo dare in diretta i seguenti ordini (chiamati **note operative**):

➤ **RUN**

che avvia l'esecuzione del programma. Quest'ordine comporta anche la cancellazione dalla memoria dei valori delle variabili eventualmente presenti.

➤ **NEW**

invece permette di cancellare dalla memoria RAM (ma non dallo schermo) tutto ciò che c'è nell'area programmi. Se invece si vuole cancellare il contenuto dell'area variabili si dovrà usare l'ordine (inseribile anche all'interno di un programma):

➤ **CLEAR**

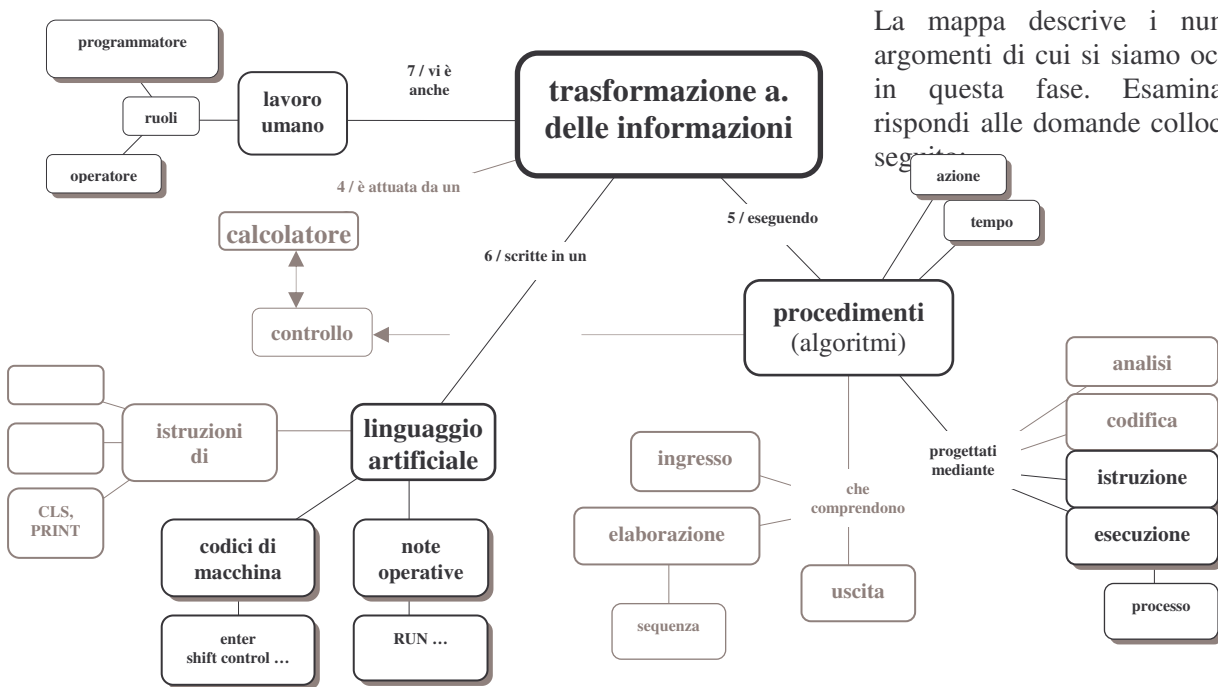
Per interrompere l'esecuzione prima che essa giunga alla conclusione dovranno essere schiacciati i tasti **Ctrl.** e **Bloc Scorr**. Per riprenderla basterà schiacciare un tasto qualsiasi.

il programma è pronto!

A questo punto il programma è stato verificato e memorizzato e potrà essere eseguito ogni volta che vi sarà quel tipo di problema da risolvere. Per poterlo utilizzare servirà un **operatore** che sia in grado di avviarlo e di assistere il calcolatore nella fase di ingresso di dati.

Il **tempo** di esecuzione dipenderà dalla velocità del microprocessore (si misura in megahertz). La sequenza delle azioni che vengono effettivamente svolte durante un'esecuzione viene chiamata **processo**. Nelle procedure sequenziali viste sino ad ora il processo coincide con il programma.

in sintesi ...



La mappa descrive i numerosi argomenti di cui si siamo occupati in questa fase. Esamina e rispondi alle domande collocate di seguito.

Quale è il ruolo del programmatore e quale quello dell'operatore

Che cosa sono e quando vengono usati i codici di macchina

Che cosa sono e quando vengono usate le note operative

altri ordini basic

In questa fase approfondiamo alcuni ordini che vediamo spesso in funzione quando il calcolatore esegue programmi professionali o giochi. Dietro di essi vi sono "routine" cioè sequenze di istruzioni in linguaggio macchina che permettono di svolgere alcune importanti operazioni matematiche oppure l'estrazione casuale di un numero o il conteggio del tempo.

Le funzioni matematiche

➤ **INT**(numero decimale o variabile contenente un numero decimale)

restituisce il numero intero immediatamente inferiore o uguale a una determinata espressione numerica

es: `PRINT INT(2.75)` darà sullo schermo il numero 2

es: `PRINT INT(-2.75)` darà sullo schermo il numero -3

anche con `A=2.75 : B=INT(A) : PRINT B` sullo schermo otterrò il numero 2

➤ **FIX**(numero o variabile numerica)

restituisce la parte intera di una espressione numerica senza la parte decimale

es: `PRINT FIX(-2.75)` darà sullo schermo il numero -2

➤ **ABS**(numero o variabile numerica)

restituisce il valore numerico assoluto (privo di segno) di una espressione numerica

es: `PRINT ABS(-12.2)` darà sullo schermo il numero 12.2

➤ **SQR**(numero o variabile numerica)

restituisce la radice quadrata di una espressione numerica (che deve essere positiva o nulla)

es: `PRINT SQR(9)` darà sullo schermo il numero 3

solo in VISUAL BASIC

➤ **Hex**(numero decimale)

restituisce il corrispondente numero esadecimale di un dato numero decimale

L'emissione di suoni

I primi calcolatori, come abbiamo già visto, erano in grado di svolgere solo calcoli, in binario, e non avevano né monitor né tastiere.

Quando iniziò a porsi il problema di ottenere una comunicazione più efficace tra calcolatore ed operatore, venne introdotto l'uso dei caratteri (grazie al codice ASCII) e dunque iniziarono ad essere utilizzati monitor e tastiere.

Oltre a poter visualizzare sul monitor le istruzioni da dare al calcolatore e permettere di ottenere su di esso i risultati dell'elaborazione, si decise di segnalare, utilizzando un'emissione di suoni, il corretto o non

corretto svolgimento delle principali fasi di caricamento del sistema operativo. Venne dunque approntato un piccolo altoparlante interno (detto “cicalino”) capace di emettere semplici suoni.

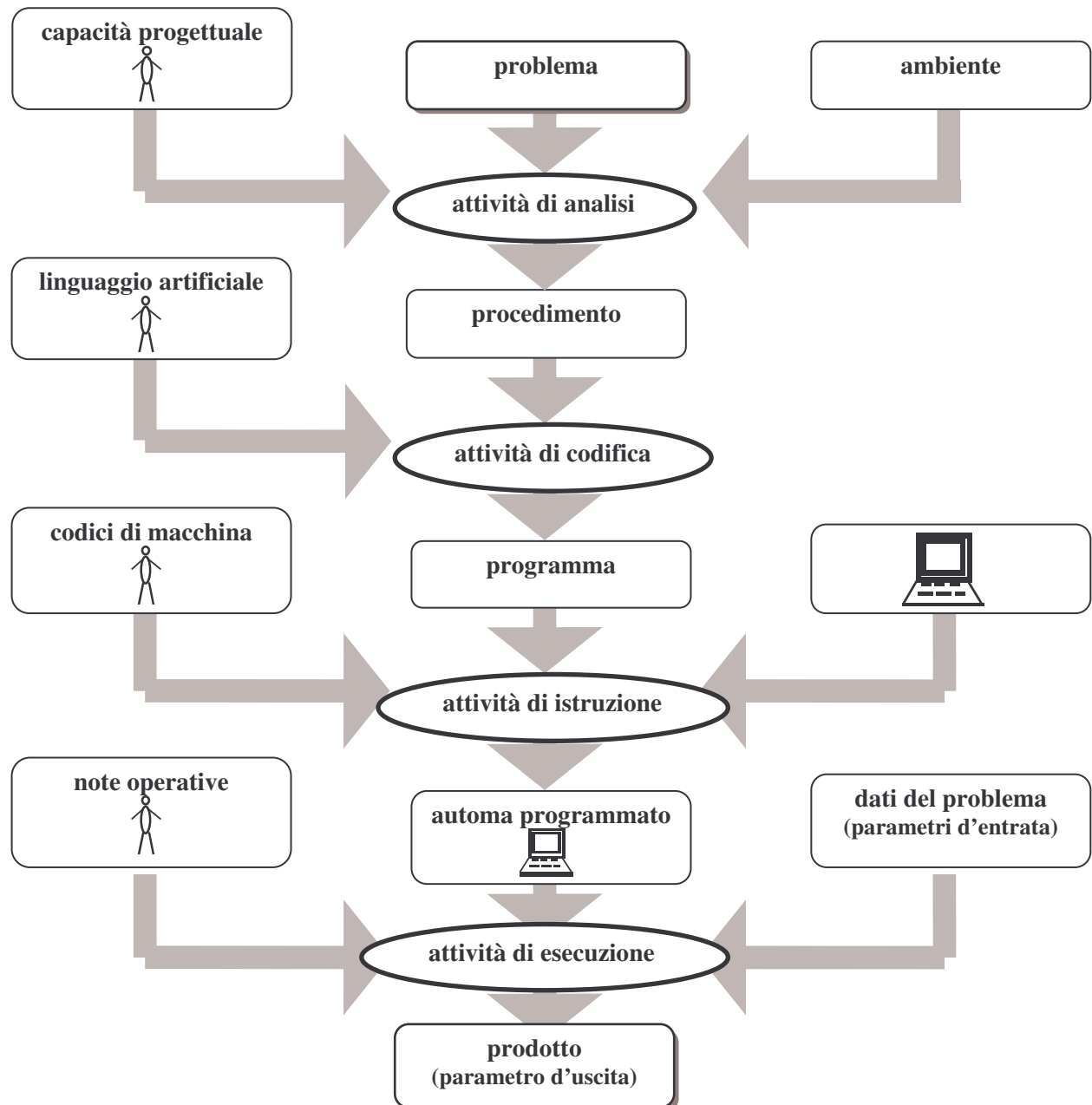
In GWBASIC l'ordine:

➤ BEEP

causa l'emissione di un suono

Dal problema al prodotto

Prima di approfondire alcune caratteristiche del linguaggio artificiale che abbiamo iniziato a studiare, esaminiamo il seguente schema che riepiloga il percorso fatto sinora:



DATI ALFANUMERICI

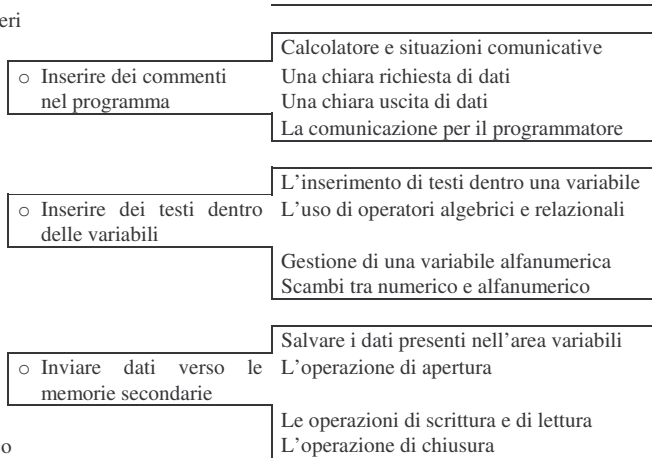
fase 3 DATI ALFANUMERICI

- nei primi calcolatori solo numeri
- grazie ai byte alfanumerici compaiono i caratteri
- l'uscita su video e stampante
- come conoscere il codice dei caratteri
- tabelle dei codici ASCII

• nei primi calcolatori solo numeri

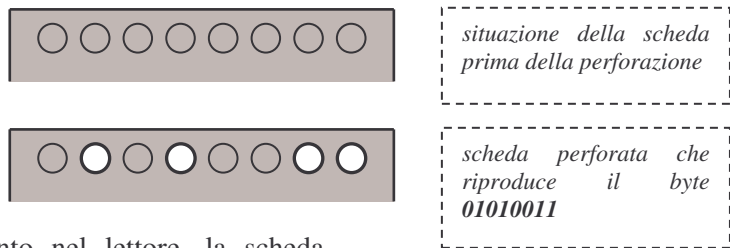
I calcolatori sono stati inventati per "fare calcoli" e dunque per elaborare numeri, opportunamente codificati in binario.

- l'organizzazione dei file su disco

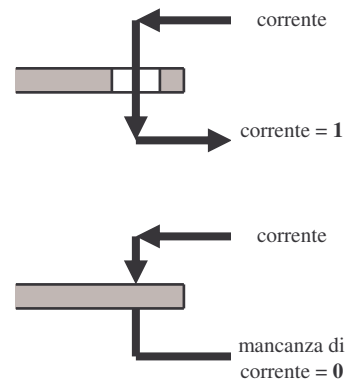


I numeri compresi tra 0 e 255 occupano un solo byte; per cifre superiori se ne dovevano aggiungere altri. I primi calcolatori erano progettati per ricevere i dati da elaborare solo in formato binario (zero ed uno) e sempre in questo formato comunicavano i risultati delle elaborazioni. Questo avveniva 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.



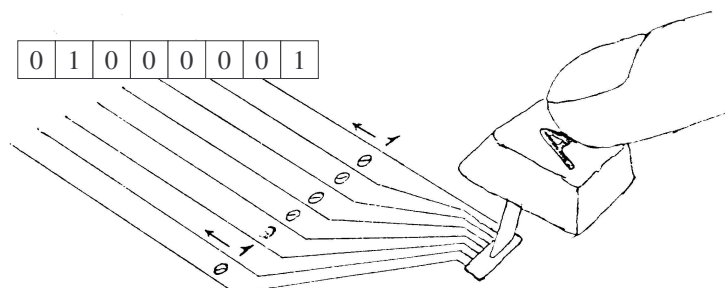
Ovviamente, prima dell'inserimento nel lettore, la scheda doveva essere forata in modo da poter riprodurre i dati da memorizzare. Questa operazione veniva fatta manualmente da appositi addetti. Nel lettore delle apposite punte metalliche mobili, collegate ad un generatore di corrente, sono situate in 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 byte alfanumerici 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

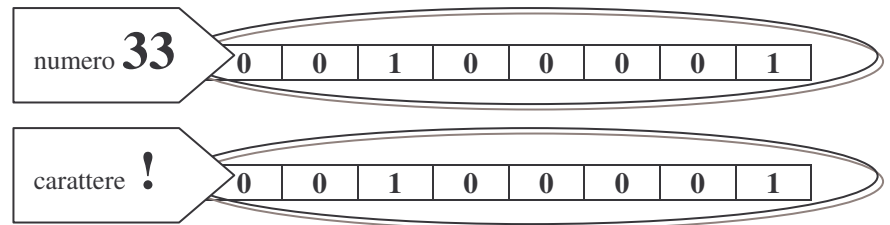


dunque studiata la possibilità di memorizzare nel calcolatore caratteri di testo, segni di punteggiatura, spazi e tutto ciò che è necessario per una corretta comunicazione.

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) devono essere contrassegnati in modo diverso 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 trentatré 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 Exchange”, 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.

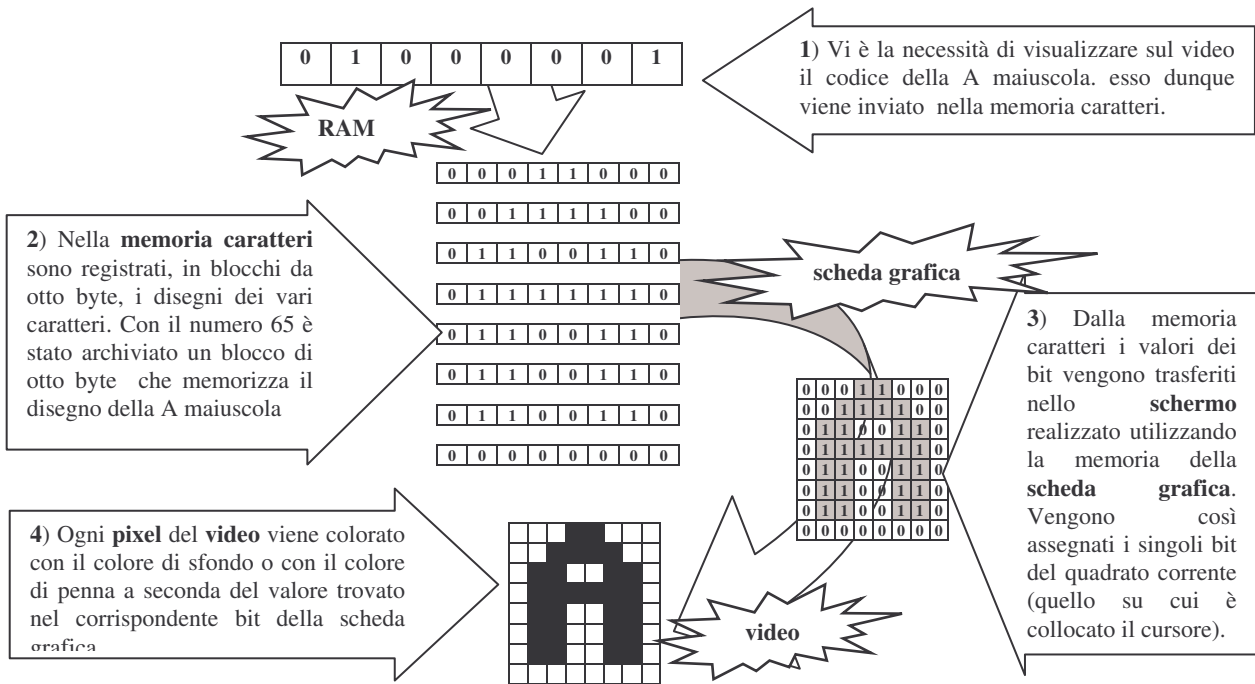
- l’uso della “memoria caratteri” per riottenere il disegno dei caratteri

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.

Vediamo ora un grafico che illustra questo percorso:



Nota bene: come abbiamo visto, le stampanti ricevono dal calcolatore i codici ASCII di ciò che devono stampare. Anch'esse possiedono una propria *memoria caratteri* che permette di ottenere su carta il disegno dei caratteri. Può capitare, stampando un testo su una stampante diversa dal solito, di ottenere i caratteri in uno stile diverso da quello scelto. Questo vuol dire che quella stampante non possiede, nella memoria caratteri, il font * precedentemente scelto.

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:

➤ **ASC(" ")** carattere
 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:

➤ **CHR\$()** numero
 Esempio: con **PRINT CHR\$(67)** otterrò sullo schermo il carattere C

- 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).

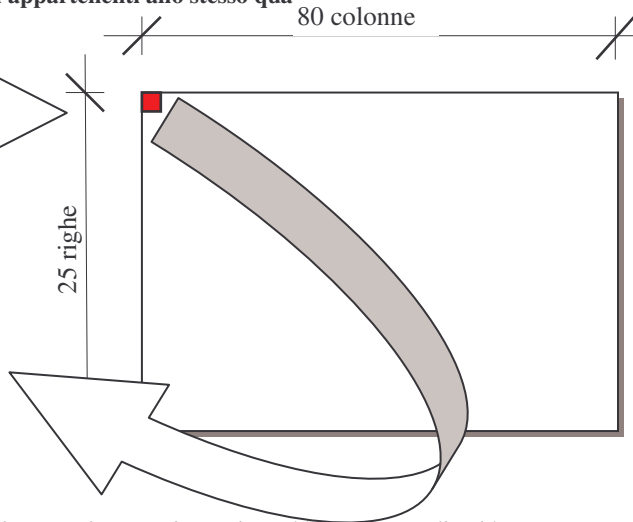
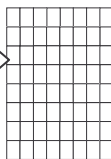
1 □	45 -	89 Y	133 ...	177 ±	221 Ý
2 □	46 .	90 Z	134 †	178 ²	222 Þ
3 □	47 /	91 [135 ‡	179 ³	223 ß
4 □	48 0	92 \	136 ^	180 ´	224 à
5 □	49 1	93]	137 ‰	181 µ	225 á
6 □	50 2	94 ^	138 Š	182 ¶	226 â
7 □	51 3	95 _	139 <	183 ·	227 ã
8 □	52 4	96 `	140 Œ	184 ´	228 ä
9	53 5	97 a	141 □	185 ´	229 å
10	54 6	98 b	142 Ž	186 °	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 ĸ	235 ë
16 □	60 <	104 h	148 ”	192 Å	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 Ě	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 Ě	247 ÷
28 □	72 H	116 t	160	204 Ì	248 ø
29 □	73 I	117 u	161 ĵ	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 Ð	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	

la modalità “testo” in MSDOS

Quando carichiamo il sistema operativo MSDOS, il calcolatore si prepara a lavorare in modalità “testo”. Il nome ci ricorda che quell’organizzazione è stata ideata apposta per poter scrivere agevolmente dei testi. Il video viene diviso in 25 righe (numerata da 1 a 25) e 80 colonne (numerata da 1 a 80). L’incrocio di righe e colonne forma dei quadrati che a loro volta sono suddivisi in una griglia di 8*8 piccoli quadratini detti pixel. Colorando i pixel con un colore (colore di penna) diverso dal colore di sfondo sarà possibile ottenere il disegno dei vari caratteri. Non è però possibile gestire singolarmente i pixel appartenenti allo stesso quadrato

1) lo schermo MSDOS in modalità testo è formato da 25 righe per 80 colonne. Vengono così formati sullo schermo 2000 “quadrati” che vengono gestiti singolarmente dal calcolatore.

2) ogni “quadrato” è formato da 64 pixel collocati in una griglia di 8 righe per 8 colonne. Colorando i pixel in modo diverso dallo sfondo sarà possibile ottenere il disegno dei vari caratteri.

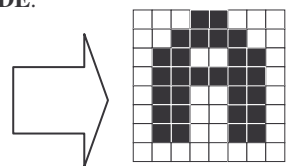


Nello schermo in modalità testo i “quadrati” utilizzati per disegnare i caratteri sono in realtà dei rettangoli. Vi è stata una compressione laterale che serve a raddoppiare il numero dei caratteri per riga (da 40 a 80).

Per far assumere ai caratteri una forma regolare è possibile utilizzare in MS-DOS l’ordine **MODE**.

Battendo **MODE 40** seguito da Invio i caratteri per riga si ridurranno a 40 ma assumeranno una forma regolare e sarà possibile individuare i vari pixel che compongono il carattere. Se, tenendo premuto il tasto SHIFT (la freccia larga rivolta verso l’alto), si schiaccia il tasto “A” sullo schermo comparirà il carattere disegnato a fianco.

Nota Bene: l’ordine **CLS**, seguito da Invio, pulirà invece lo schermo.



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
- memorizzati, insieme al programma, *come dati alfanumerici* ad uso del calcolatore oppure come messaggi rivolti all’operatore
- 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

Dati alfanumerici nel programma

La comparsa di tastiera e monitor ha permesso ai programmatori di scrivere i programmi non più in linguaggio macchina ma utilizzando linguaggi di programmazione più vicini al linguaggio naturale dell'uomo (in particolare la lingua inglese).

In questo modo le istruzioni, scritte e memorizzate in caratteri alfanumerici, possono essere facilmente visualizzate sul monitor. Saranno tradotte in **linguaggio macchina** solo in un secondo momento, grazie ad un **programma di compilazione**.

Oltre a questa possibilità, la presenza di caratteri sulla tastiera ha permesso ai programmatori l'inserimento di **messaggi**:

- rivolti all'**operatore** (l'utente che utilizzerà l'applicazione prodotta del programmatore)
- rivolti ad altri **programmatori** che saranno chiamati a modificare il software prodotto.

calcolatore e situazioni comunicative

Esaminiamo le **situazioni comunicative** che si sono realizzate sinora nella nostra attività con il calcolatore.



Quando questi programmi saranno utilizzati, davanti al calcolatore non vi sarà il programmatore ma un **operatore** (utente) che spesso non ha alcuna conoscenza di informatica e del programma che sta utilizzando.

Per realizzare la massima chiarezza su ciò che viene chiesto all'operatore, il programmatore dovrà inserire negli ordini che gestiscono la comunicazione con l'operatore (INPUT e PRINT) dei testi rivolti all'operatore.

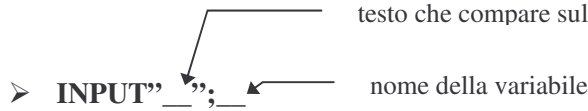
Si realizzerà pertanto la seguente situazione comunicativa:



Tale situazione comunicativa sarà gestita dal programma e sarà dunque compito del programmatore, in base al tipo di destinatari a cui esso è destinato, garantire la massima completezza nella comunicazione. Sarà, ad esempio, differente il linguaggio utilizzato dal calcolatore che opera una banca da quello che opererà nello studio professionale di un medico.

Va comunque tenuto conto che l'introduzione del mouse, affermatosi in modo generalizzato alla fine degli anni 80', e del successivo sistema operativo Window, ad esso collegato, hanno modificato in modo radicale le modalità di comunicazione tra calcolatore e utente. Per ora vediamo come operare con il vecchio GWBASIC in ambiente MS-DOS.

La realizzazione di una chiara richiesta di dati

Con: 

E' possibile far comparire sul video una scritta che spieghi cosa il calcolatore sta richiedendo.

Esempio:

Invece di scrivere nel programma: 10 INPUT A

si potrà scrivere: 10 INPUT "Scrivi il prezzo di vendita al chilo "; Pvend

La realizzazione di una chiara uscita di dati

Anche dopo l'ordine PRINT sarà possibile inserire tra virgolette una spiegazione che potrà precedere e anche seguire il nome della variabile contenente il nome del dato in uscita.

Esempio: 100 PRINT "La paga base è di euro "; A;" mensili"

Potremo ora modificare i programmi sinora realizzati rendendo più comprensibile la comunicazione tra calcolatore ed operatore. Carichiamo dunque il primo programma, quello del venditore di arance, e trascriviamo qui sotto le istruzioni di ingresso e di uscita:

INPUT Pvend	INPUT "scrivi il prezzo di vendita al chilo"; Pvend
INPUT	INPUT "..
INPUT	INPUT "..
INPUT	INPUT "..
.....	
PRINT	PRINT "..

Dopo aver concordato con il gruppo di lavoro le modifiche da apportare, potremo inserirle nelle linee di programma. Per attivare la funzione di inserimento collocheremo il cursore dove deve essere inserito il testo e schiacteremo **Ctrl.** e **R.**

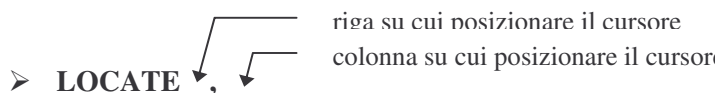
Il cursore diventa un rettangolino bianco intermittente e ciò che batteremo ora sulla tastiera non si sovrappone ma sposta il testo che segue.

Bisogna ricordare che le correzioni saranno memorizzate nella RAM solo quando batteremo  .

Provato il programma lo potremo salvare con il nome precedente. In questo modo la vecchia versione sarà sostituita da quella contenente le modifiche apportate.

Oltre ai linguaggi e ai termini utilizzati, per ottenere una valida comunicazione sarà importante la gestione grafica dello schermo.

E' possibile collocare il cursore in una determinata posizione dello schermo, pronto per la scrittura. Lavorando con il GW_BASIC, si utilizzerà:



Ad esempio:

LOCATE 5,10:PRINT" Laura"

colloca il nome Laura sullo schermo. Il primo carattere sarà sulla colonna 5 e sulla riga 10

Nota Bene: lo schermo GW_BASIC dispone di 25 righe (numerata da 1 a 25) e di 80 colonne (numerata da 1 a 80). La venticinquesima riga sarà però utilizzabile solo se ripulita dai codici dei tasti funzione mediante l'ordine:

➤ **KEY OFF**

La comunicazione per il programmatore

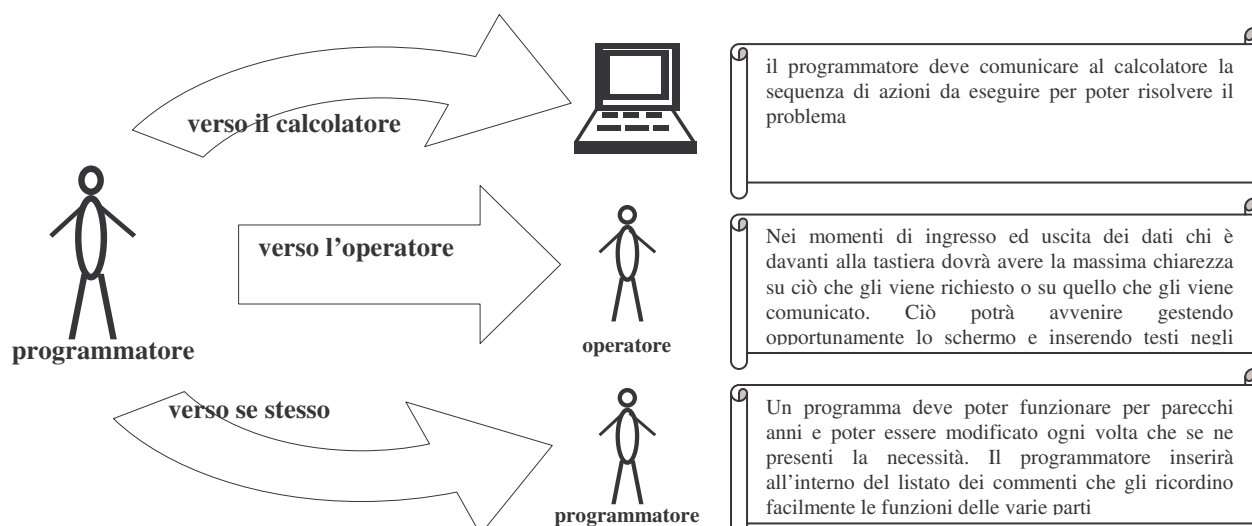
Più i programmi sono complessi più sarà facile per lo stesso programmatore dimenticare la funzione di alcune parti. Inoltre i programmi vanno spesso rivisti e la revisione sarà ancora più problematica se non verrà effettuata dallo stesso programmatore che ha realizzato il programma.

Per ovviare a questi problemi è buona abitudine inserire nel programma dei commenti che compariranno solo nel listato del programma. Essi devono essere preceduti dall'istruzione:

➤ **REM**

Ad esempio: 50 LET Kvend=Kacq-Kscart: REM calcolo del prezzo di vendita al pubblico delle arance
Nel momento dell'esecuzione il calcolatore, dopo aver eseguito le istruzioni all'inizio della linea 50, ignorerà tutto ciò che segue l'ordine REM proseguendo l'esecuzione alla linea successiva. Per questo motivo sulla parte di linea che segue il REM non vanno collocati altri ordini per il calcolatore. REM può essere sostituito dall'apostrofo ('). Tutto questo vale anche per il VISUAL BASIC.

Nell'attività di programmazione possiamo individuare tre tipi di comunicazione:



Riscriviamo di nuovo il "programma per il venditore di arance" apportandovi le seguenti modifiche:

Ogni ingresso di dati dovrà sostituire il precedente, apparire al centro dello schermo e utilizzare un diverso colore di sfondo. La stessa cosa dovrà avvenire per l'uscita finale. Ogni elaborazione dovrà essere commentata.

In VISUAL BASIC

Da questa fase iniziamo a studiare, sui vari argomenti affrontati, anche le istruzioni che ci mette a disposizione il VISUAL BASIC (VB) in modo da essere pronti, quando inizieremo ad utilizzare questo linguaggio, a sfruttarne tutte le possibilità.

Anche se il VB deriva sostanzialmente dal GWBASIC esistono alcune **differenze** che esamineremo con cura.

Sono inoltre numerose ed utili le **espansioni del linguaggio** che permettono di ottenere con un solo ordine ciò che in GWBASIC si ottiene con lunghe sequenze di istruzioni.

differenze: la collocazione dei dati sullo schermo

In Windows non esiste la separazione tra modalità di testo e modalità grafica. La collocazione sulle form di testi ed oggetti avviene utilizzando un sistema di coordinate simili a quelle grafiche del Gwbasic.

Per il sistema di coordinate di Visual Basic sono valide le seguenti regole:

- Nelle istruzioni per collocare dati sullo schermo le vengono sempre espresse in **twip** *.
- L'angolo superiore sinistro dello schermo, come di qualsiasi altra finestra collocata su di esso, viene sempre rappresentato con (0, 0).

*** Utilizzo dei twip**
 Per impostazione predefinita, per tutte le istruzioni di spostamento, ridimensionamento e disegno di elementi grafici di Visual Basic viene utilizzato come unità di misura il twip. Un *twip* equivale a 1/20 di punto di stampante, ovvero 1.440 twip equivalgono a un pollice, mentre 567 twip a un centimetro. Queste misure determinano le dimensioni di un oggetto in stampa. Le distanze effettive visualizzate sullo schermo variano in base alla dimensione dello schermo.

differenze: l'ingresso dei dati dall'operatore

In VB non esiste l'ordine Input. Come vedremo, l'ingresso dei dati dall'operatore può avvenire in vari modi. Il sistema più utilizzato è quello di organizzare un'apposita finestra che viene attivata con l'ordine **InputBox**. Questo ordine permette di visualizzare, all'interno di un riquadro, tutte le informazioni utili affinché l'operatore risponda correttamente alla richiesta di dati. Questa è la sua sintassi:

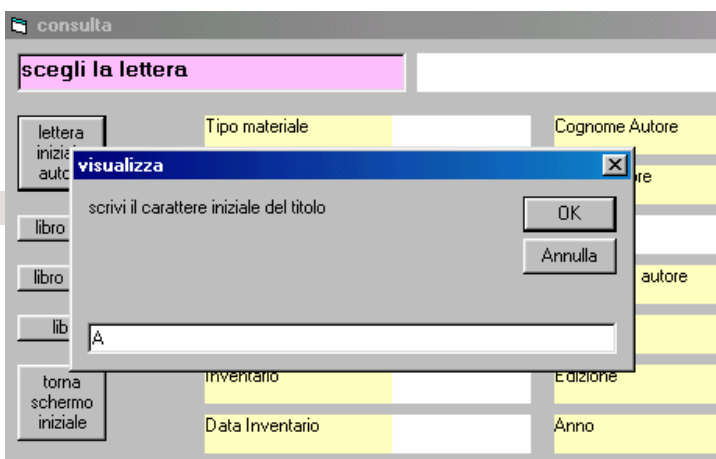
```
variabile = InputBox(prompt, titolo, predefinito, posx, posy)
```

Parte	Descrizione
<i>variabile</i>	Nome della variabile dentro la quale verrà inserito quanto digitato dall'utente
<i>prompt</i>	Espressione di stringa che costituisce il messaggio visualizzato nella finestra di dialogo. La lunghezza massima consentita per l'argomento <i>prompt</i> è pari a circa 1024 caratteri e dipende della larghezza dei caratteri utilizzati.
<i>titolo</i>	Espressione stringa visualizzata nella barra del titolo della finestra di dialogo. Se si omette l'argomento <i>titolo</i> , nella barra del titolo verrà indicato il nome dell'applicazione.
<i>predefinito</i>	Espressione stringa visualizzata nella casella di testo come risposta predefinita se non viene fornito alcun input. Se si omette l'argomento <i>predefinito</i> , verrà visualizzata una casella di testo vuota.
<i>posx</i>	Espressione numerica che specifica, in twip, la distanza orizzontale tra il bordo sinistro della finestra di dialogo e il bordo sinistro dello schermo. Se si omette l'argomento <i>posx</i> , la finestra di dialogo risulterà centrata orizzontalmente.
<i>posy</i>	Espressione numerica che specifica, in twip, la distanza verticale tra il bordo superiore della finestra di dialogo e il bordo superiore dello schermo. Se si omette l'argomento <i>posy</i> , la finestra di dialogo risulterà centrata verticalmente a circa un terzo dal bordo superiore dello schermo.

Esempio nel programma per la gestione della biblioteca scolastica l'esecuzione dell'ordine:

```
scrivi = InputBox("scrivi il carattere iniziale del titolo", "visualizza", A, 2400, 2600)
```

provoca l'uscita della finestra di input collocata a fianco:



Per l'uscita di dati potrà invece essere utilizzato l'ordine **Print** che però spesso viene sostituito da una finestra attivabile con l'ordine **MsgBox** e funzionante in modo simile a InputBox. Inoltre in VB al posto di Locate si utilizzano due ordini (**Current x** e **Current y**) che fanno riferimento direttamente ai pixel di schermo in orizzontale (x) e in verticale (y) a partire dall'angolo superiore sinistro dello schermo.

Ad esempio con Current x = 2500 : Current y = 500 si stabilisce di collocare il cursore spostato a destra di 2500 pixel e in basso di 500 pixel rispetto all'angolo superiore sinistro dello schermo

Ed ora al lavoro!

Qui di seguito vi sono alcuni problemi di matematica. 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. Dovrai fare correttamente queste operazioni:

- trasformare il problema di matematica in problema di informatica mantenendo costanti i dati segnalati e rendendo parametrici gli altri
- disegnare il grafo di scomposizione
- compiere l'analisi dei dati
- realizzare il programma curando la comunicazione diretta all'operatore e quella diretta al programmatore, sia in fase di ingresso dati che in uscita
- compiere, dopo aver istruito il calcolatore, l'esecuzione di prova controllando la correttezza dei risultati

Nota Bene > alcuni dati devono mantenere il loro valore costante

13) Carlo acquista **tre 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 a Carlo?

14) Un proprietario terriero deve trasportare **32 tonnellate di patate** al mercato ortofrutticolo e dispone di **2 autocarri** della portata di **40 quintali ciascuno** (da mantenere come dato costante). Quanti viaggi deve fare ogni autocarro per trasportare tutte le arance?

15) La vecchia edizione di un libro di testo è composta da **450 pagine**. Nella nuova edizione dello stesso testo vengono aggiunti **3 capitoli di 25 pagine l'uno** ed eliminati **2 capitoli di 20 pagine ciascuno**. Quante pagine ha il testo della nuova edizione?

16) I partecipanti ad una gita prendono posto su **6 pullman** e **10 automobili**. Su ciascun pullman **i posti sono 42** e quelli di **ogni automobile 5** (da mantenere come dato costante). Quanti sono complessivamente i gitanti se non rimangono posti vuoti?

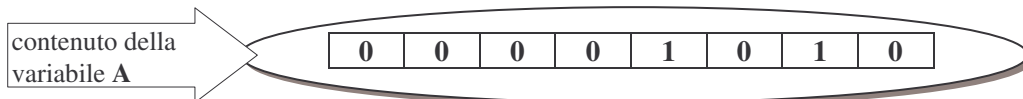
Le variabili alfanumeriche o "stringa"

L'inserimento di testi dentro una variabile

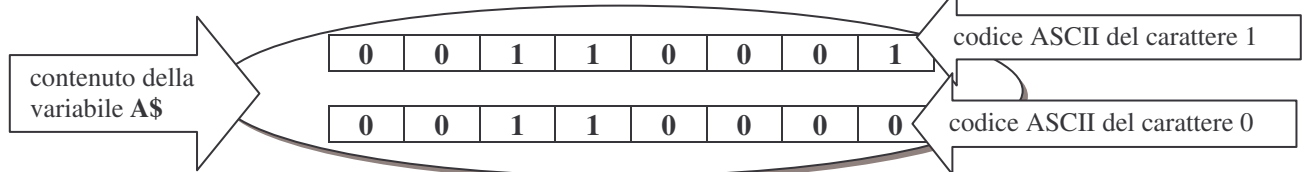
I byte che codificano i caratteri presenti sulla tastiera, oltre che ad essere utilizzati nel programma per inserire commenti rivolti all'operatore o allo stesso programmatore, possono essere inseriti dentro delle apposite variabili. In questo caso la variabile si chiama **alfanumerica** o "stringa" e in quest'ultimo modo viene denominato anche il suo contenuto. Essa 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 **invio**, il calcolatore organizzerà nella sua RAM una variabile numerica che occuperà un byte e sarà così strutturata:

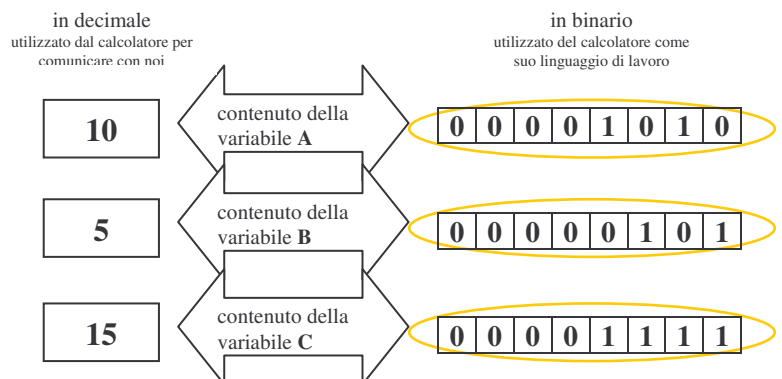


Mentre se batterò **A\$="10"** seguito da **invio**, il calcolatore organizzerà nella sua RAM una variabile alfanumerica che occuperà due byte e sarà così strutturata:

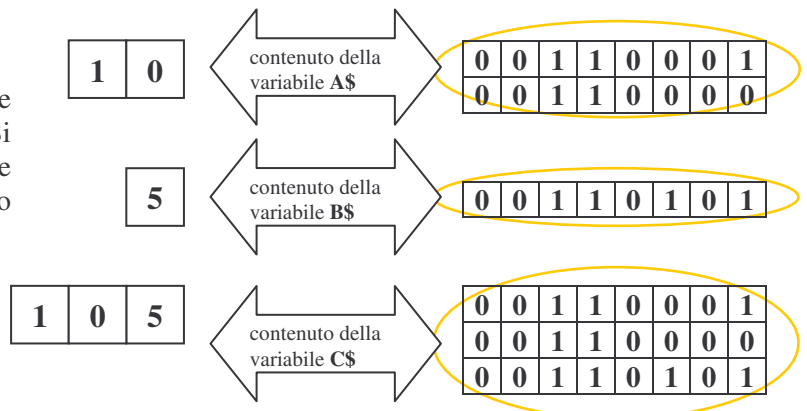


Di conseguenza:

Se ad esempio A=10 e B=5 se assegno C=A+B dentro la variabile C sarà collocato il numero 15.



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!

ATTENZIONE! In GWBASIC una variabile alfanumerica può ospitare sino ad un massimo di 255 caratteri e occupa tanti byte quanti sono i suoi caratteri.
 Ad una variabile numerica basterà un byte per ospitare numeri sino al 255:

La somma dei valori presenti nei singoli bit è 255

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

Per valori superiori bisognerà utilizzare i bit di un secondo byte

Ora ad esempio la somma è di 260

0	0	0	0	0	0	0	0	256
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	1	0	0

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 GWBASIC 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 "" (stringa 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.

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

↑ nota lo spazio in coda

Come approfondiremo nell'unità PROGRAMMI, possono invece essere usati tutti gli operatori relazionali (maggiore, minore, uguale, diverso, ecc...). In questo caso il calcolatore prende in considerazione i codici ASCII dei singoli caratteri.

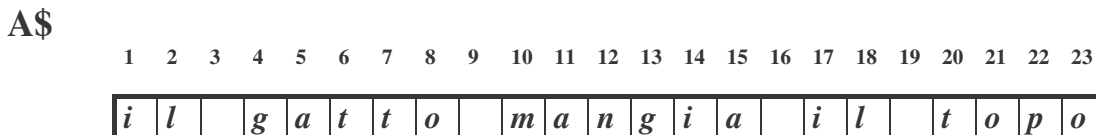
La struttura della variabile alfanumerica

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.

Ad esempio dopo l'esecuzione dell'ordine:

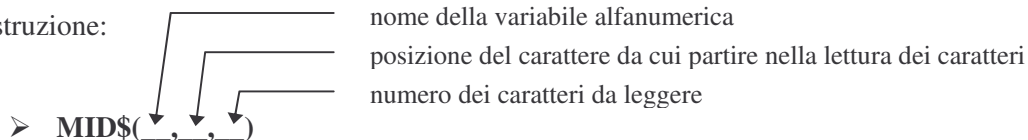
A\$="il gatto mangia il topo"

dentro la memoria del calcolatore la variabile A\$ sarà così strutturata:



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:



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".*

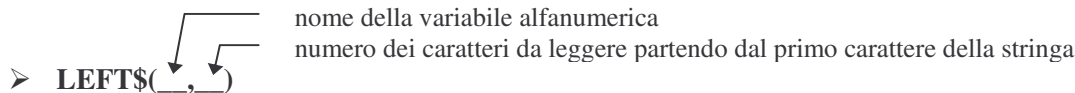
Mentre se non scriviamo il numero dei caratteri da leggere il calcolatore prosegue sino alla fine della stringa:

*con **MID\$(A\$,17)** otterrò la parola "il topo".*

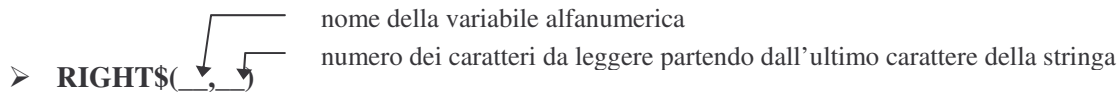
Quest'ordine rende possibile riassegnare solo una parte di una variabile stringa.

*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:

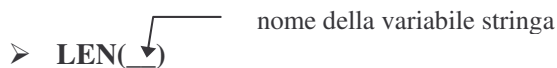


*Esempio: con **LEFT\$(A\$,10)** otterrà la parola "il gatto m"*



*Esempio: con **RIGHT\$(A\$,7)** otterrà la parola "il topo"*

Inoltre l'istruzione:



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

*Esempio: con **PRINT LEN(A\$)** il calcolatore scriverà sullo schermo il numero 23*

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:

➤ $\text{VAL}(\underline{\quad})$ nome o contenuto di una variabile stringa

Essa permette di utilizzare per operazioni matematiche il numero contenuto in una variabile stringa.

Ad esempio se: $\text{B\$}="4"$ con $\text{A}=2*\text{VAL}(\text{B\$})$ potrò utilizzare quel carattere come numero per fare una moltiplicazione.

Se dopo aver svolto l'esempio precedente battete l'ordine $\text{PRINT LEN}(\text{B\$})$ il calcolatore scriverà sullo schermo 2 e non 1 come sarebbe logico visto che B\$ contiene il numero 4 (come carattere).

La spiegazione è nel fatto che il calcolatore quando opera un passaggio da numerico ad alfanumerico colloca sempre uno spazio prima del numero (la stessa cosa succede quando viene scritto un numero sullo schermo). Se si desidera avere nella variabile alfanumerica solo il carattere che codifica il numero andrà operato uno slicing.

Ad esempio con $\text{B\$}=\text{mid}(\text{b\$},2)$ riassegno B\$ escludendo il primo carattere

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

➤ $\text{STR}(\underline{\quad})$ nome o contenuto di una variabile

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

Esempi: $\text{A}=15:\text{B\$}=\text{STR}(\text{A})$ o $\text{B\$}=\text{STR}(15)$ o anche $\text{C\$}=\text{STR}(5)+\text{STR}(\text{A})$

In VISUAL BASIC

differenze

in VB non è necessario, ma è comunque possibile, far seguire il nome della variabile dal carattere speciale \$. Anche nelle istruzioni il carattere \$ è facoltativo ($\text{mid}\$=\text{mid}$).

Il calcolatore è comunque in grado di individuare come testo ciò che viene inserito in memoria e di trattarlo conseguentemente. Inoltre la variabile **string** in VB può contenere sino a 4.294.967.296 (2^{31}) caratteri consentendo di memorizzarvi interi libri e di realizzare applicazioni di trattamento testi.

espansioni del linguaggio

Tra i codici ASCII dei caratteri maiuscoli e quelli dei caratteri minuscoli esiste una differenza costante di 32 ($A>65, a>97$). Infatti se eseguiamo in GWBASIC:

Input "scrivi un carattere maiuscolo";a\$

a=asc(a\$) : b=a+32

Print "carattere maiuscolo ";chr\$(b)

il calcolatore, dato un carattere maiuscolo, ne scriverà il corrispondente carattere minuscolo.

In un programma può essere utile rimuovere questa distinzione facendo sì che ad un controllo "paolo" risulti uguale a "PAOLO". La seguente istruzione VB, dal momento in cui viene eseguita, permette di eliminare la distinzione tra maiuscole e minuscole

- **Option Compare Text** permette di non effettuare alcuna distinzione.
 - **Option Compare Binary** ripristina la differenziazione
- inoltre**
- &** > restituisce una stringa che è la concatenazione di altre due (si aggiunge a + vista in GWBASIC). *esempio "Buongiorno" & "signori" restituisce "Buongiorno signori"*
- Instr** > restituisce la posizione di una sottostringa di caratteri in una stringa di caratteri *esempio Instr("Mio papa è là", "pa") restituisce 5 (primo "pa")*
- LTrim** > restituisce una stringa di caratteri da cui vengono eliminati tutti gli spazi situati a sinistra di una determinata stringa di caratteri
- RTrim** > restituisce una stringa di caratteri da cui vengono eliminati tutti gli spazi situati a destra di una determinata stringa di caratteri
- Trim** > restituisce una stringa di caratteri da cui vengono eliminati tutti gli spazi situati a sinistra e a destra di una determinata stringa di caratteri
- Lcase** > converte in minuscolo tutti i caratteri alfabetici di una stringa di caratteri
- Ucase** > converte in maiuscolo tutti i caratteri alfabetici di una stringa di caratteri

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	I	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\$= _____

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 _____

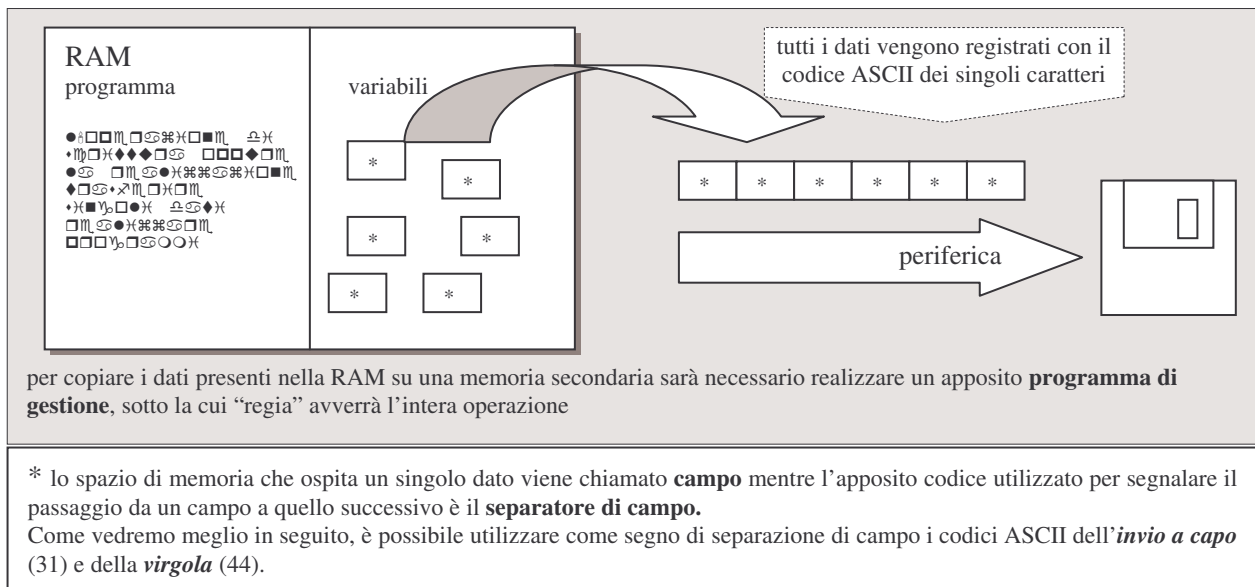
L’invio di dati verso le memorie secondarie

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), 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. Insieme ai dati inseriti nell’area 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 un drive (ma anche verso un registratore o una stampante) 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, per quanto riguarda drive e nastri, 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.



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.

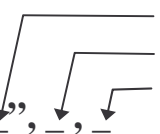
E' dunque necessario "avvisare" il calcolatore che si intende inviare dei dati presenti nella sua memoria RAM verso una memoria secondaria sulla quale saranno collocati sequenzialmente oppure che si intende leggere una sequenza di dati da una memoria secondaria per collocarli dentro delle variabili organizzate 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 drive verso il quale vengono indirizzati i dati qualora esso sia diverso dal drive 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 GWBASIC si usa l'ordine:

➤ **OPEN** " ", , ,



direzione dei dati (I=ingresso O=uscita)
 numero del canale (da 1 a 10)
 nome eventualmente preceduto da drive e percorso

esempio: con OPEN "I",2,"b: alunni\92"

viene aperto per la lettura un file chiamato 92 e collocato sulla directory alunni del disco collocato nel drive b. Come numero del canale viene scelto il 2.

Quando riceve l'ordine di apertura per la scrittura il calcolatore controlla se il drive richiesto è collegato e se è presente il dischetto. Verifica inoltre se su dischetto vi sono ancora blocchi liberi e sposta la testina 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 il calcolatore controlla se il drive richiesto è collegato e se è presente il dischetto. Verifica inoltre se sul dischetto è presente un file registrato con il nome richiesto. In caso di risposta positiva sposta la testina dalla traccia centrale 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\$="alumni" con OPEN "I",1,C\$ apriremo, per leggere, un file di nome alumni 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 fase 8 pag 31) **OPEN "I",1,"a:"+C\$**

Inoltre se il nome del file da aprire proviene da una variabile numerica è importante tener conto di quanto detto nella fase 8 pag. 33 (scambi tra numerico e alfanumerica)

la lettura dal supporto

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

➤ **INPUT#** , numero del canale
variabile che dovrà ospitare il dato

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.

➤ **LINE INPUT#** , numero del canale
variabile che dovrà ospitare il dato

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 il 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:

➤ **CLOSE** numero del canale

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:

```
5 A$ = "gatto": B = 27
10 OPEN "O",1,"c: dati.txt"
20 WRITE#1,A$,B
30 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 €)*

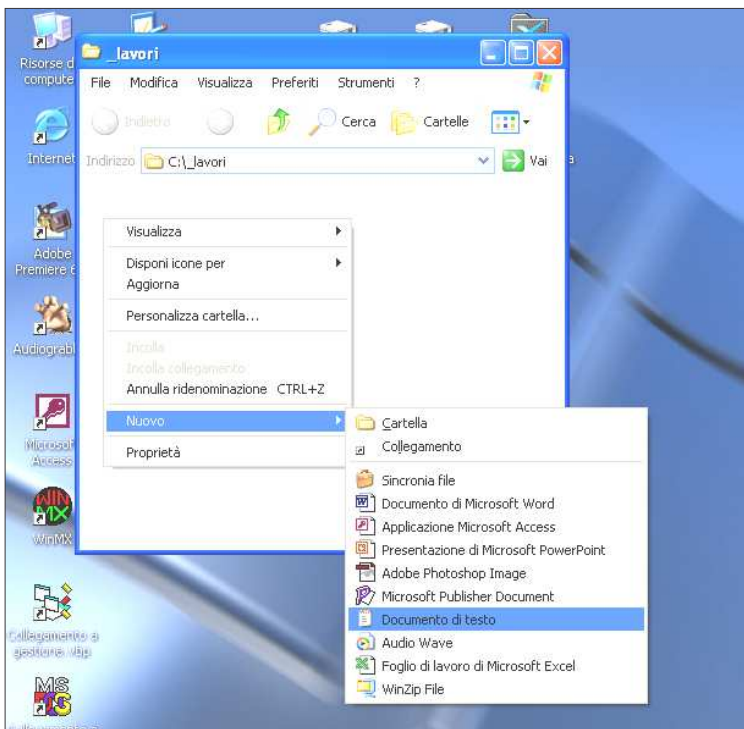
Problema di matematica

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?

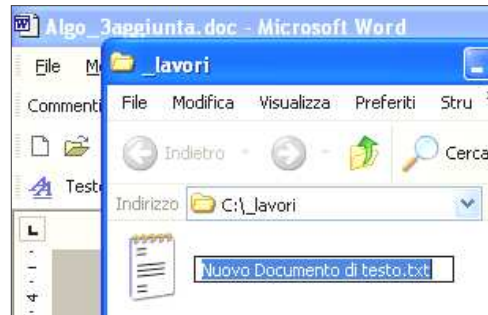
Problema di informatica

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.



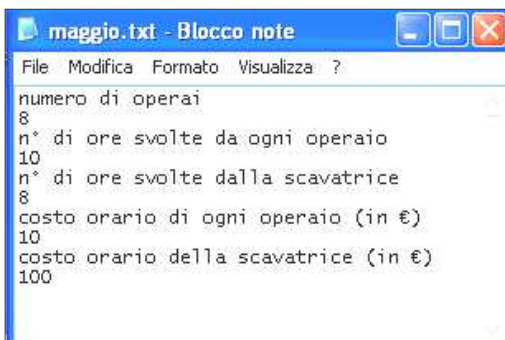
Per creare un primo file di dati apriamo la cartella **di gruppo** ed apriamo un file di testo



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 GWBASIC inizieremo a scrivere il programma.


```

GWPBASIC.EXE
LIST
10 REM ---- programma per calcolare, dato il mese dei lavori, il costo
20 CLS:INPUT"scrivi il nome del mese dei lavori(es:maggio)";ML$
30 REM ---- apertura del file, caricamento dei dati in RAM
40 OPEN"i",1,"c:\verde uno\"+ML$+".txt"
50 INPUT#1,NOP$,NOP,NOOP$,NOOP,NOSC$,NOSC,COOP$,COOP,COSC$,COSC
60 CLOSE 1
70 REM ---- procedura di calcolo
80 LET TOTCOOP=COOP*NOP*NOOP
90 LET TOTSCAV=NOSC*COSC
100 LET COSTLAV=TOTCOOP+TOTSCAV
110 REM --- uscita dei dati
120 CLS:PRINT NOP$; NOP
130 PRINT NOOP$; NOOP
140 PRINT NOSC$; NOSC
150 PRINT COOP$; COOP
160 PRINT COSC$; COSC
170 PRINT:PRINT "il costo dei lavori è di euro";COSTLAV
180 ENB
OK
save"c:\_lavori\scauo.bas",a
OK

```

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.

```

10 REM ---- programma per calcolare, dato il mese dei lavori, il costo
20 CLS:INPUT"scrivi il nome del mese dei lavori(es:maggio)";ML$
30 REM ---- apertura del file, caricamento dei dati in RAM
40 OPEN"i",1,"c:\verde uno\"+ML$+".txt"
50 INPUT#1,NOP$,NOP,NOOP$,NOOP,NOSC$,NOSC,COOP$,COOP,COSC$,COSC
60 CLOSE 1
70 REM ---- procedura di calcolo
80 LET TOTCOOP=COOP*NOP*NOOP
90 LET TOTSCAV=NOSC*COSC
100 LET COSTLAV=TOTCOOP+TOTSCAV
110 REM --- uscita dei dati
120 CLS:PRINT NOP$; NOP
130 PRINT NOOP$; NOOP
140 PRINT NOSC$; NOSC
150 PRINT COOP$; COOP
160 PRINT COSC$; COSC
170 PRINT:PRINT "il costo dei lavori è di euro";COSTLAV
180 _____
190 _____
200 _____
210 _____
220 _____

```

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

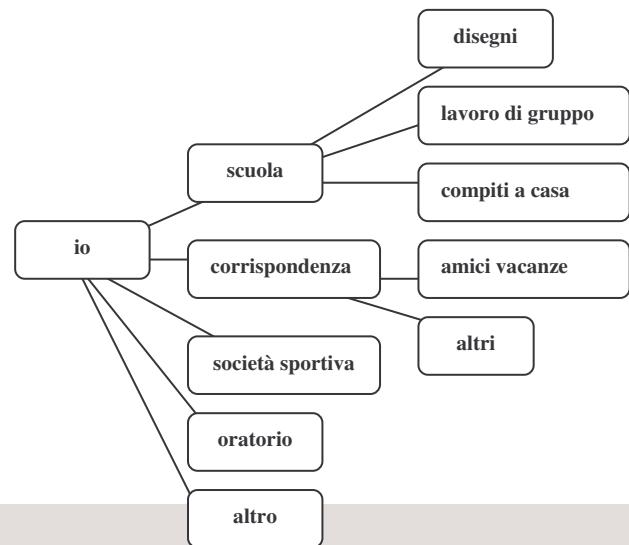
L'organizzazione dei file su disco

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, mi troverei 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 file rappresentano le foglie e i nodi (chiamati **cartelle** o **directory**) hanno la funzione di condurre ad esse, stabilendo legami logici tra quelle che dipendono dallo stesso nodo. Mentre i file vengono utilizzati per depositare archivi di 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.

A 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 GWBASIC utilizzando i seguenti ordini (validi anche in VISUAL BASIC):

➤ **MKDIR** "  " eventuale percorso e nome

viene creata una nuova directory.

Ad esempio con MKDIR "gite/adda", pur restando sulla directory principale, il calcolatore organizza una nuova directory chiamata adda collegata alla directory gite.

Per cancellare una directory bisogna prima accertarsi che sia vuota:

➤ **RMDIR** "  " eventuale percorso e nome

Inizialmente la directory corrente è la directory principale del disco che è anche la radice dell'albero. Il percorso indica i nodi da attraversare (separati tra loro dal segno "\") per spostarsi dalla directory corrente ad una nuovo nodo. Il nome della directory corrente non va indicato nel percorso nel quale non dovrà essere collocato nessuno spazio bianco.

Ad esempio con CHDIR="C:gite/resinelli" viene reso operativo l'hard disk e la directory corrente diventa resinelli.

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 directory 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 "O",1,"C:alunni\3" aprirà un file chiamato 3 sulla directory alunni dell'hard disk

mentre con LOAD "C:programmi\dati_al" caricherà il programma dati_al collocato sulla directory programmi dell'hard disk.

In VISUAL BASIC

differenze

anche se contiene gli stessi elementi visti in GWBASIC, l'apertura e la chiusura in VB presentano qualche differenza di sintassi.

*Ad esempio con: **Open "c:\programmi\alunni\prima" For Input As 1** si apre per leggere (**Input**) un file chiamato **prima**, collocato su *c:\programmi\alunni*, fissando **1** come numero di canale. Se avessimo aperto per scrivere avremmo sostituito **Input** con **Output**.*

Per la lettura, scrittura e chiusura tutto rimane invariato

espansioni del linguaggio

ChDir > permette di modificare la directory corrente

*Ad esempio con **ChDir "c:\alunni"** di renderà operativa la directory alunni su c*

ChDrive > permette di modificare l'unità disco corrente

*Ad esempio con **ChDrive "a"** si potrà rendere di nuovo operativo il dischetto*

CurDir > permette di conoscere la directory corrente per il drive di default (*CurDir*) o per un drive indicato *es CurDir("A")*

Name > permette di rinominare un file in una directory, di rinominare una directory in un disco, o di spostare un file da una directory all'altra nella stessa unità disco

*Ad esempio con **Name "h:\dir" As "h:\dirdir"** rinomina la directory*

*Ad esempio con **Name "h:\file" As "h:\dirdir\file"** rinomina la directory*

FileCopy > copia un file

*Ad esempio con **FileCopy "a:toto.txt", "c:\rep\titi"** copia il file titi dall'hard disk sul floppy rinominandolo*

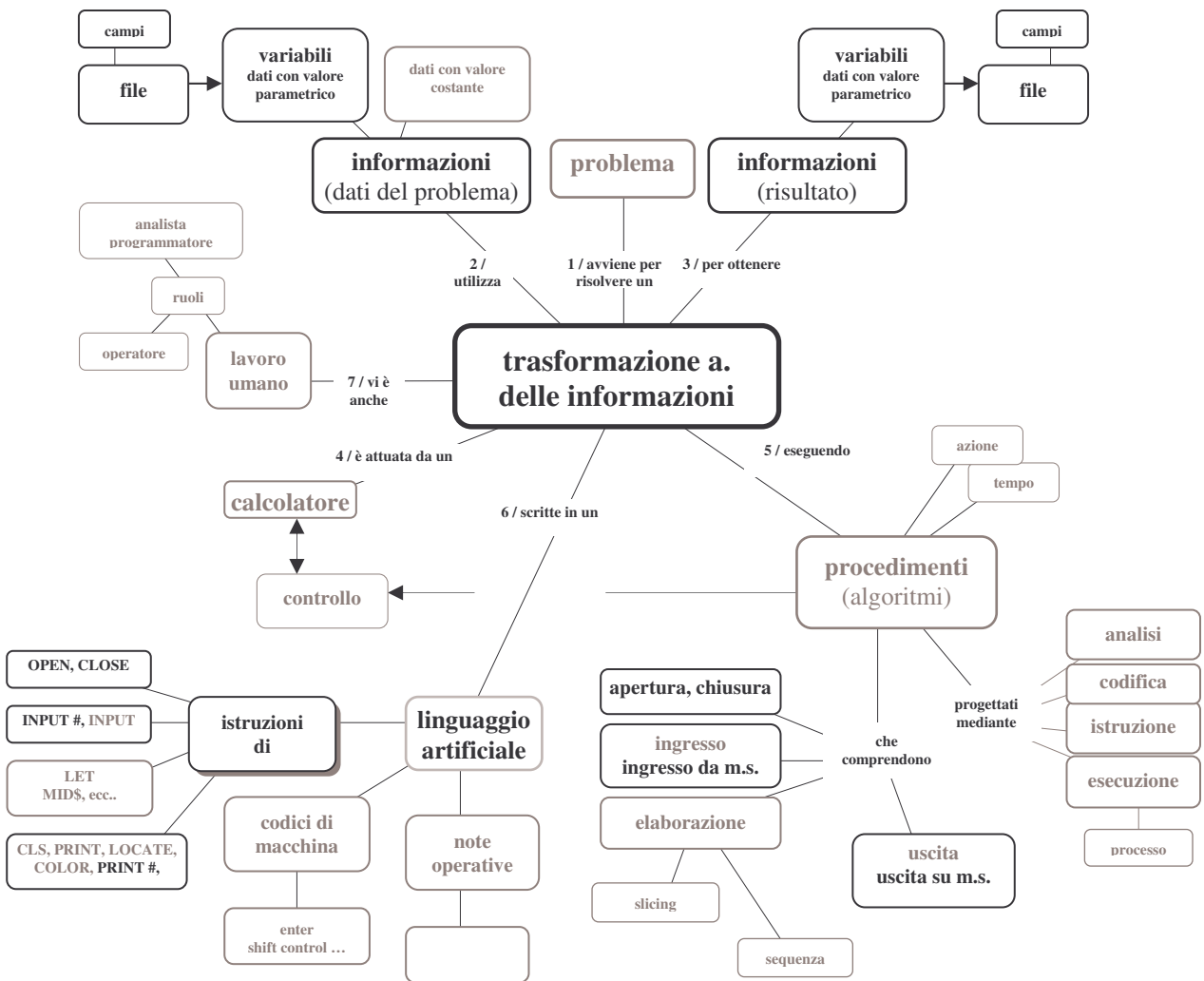
FileDateTirne > restituisce la data e l'ora dell'ultimo accesso al file

*Ad esempio con **c\$=FileDateTime("toto.txt")** viene collocata in c\$ data ed ora dell'ultimo accesso al file *toto**

FileLen > restituisce la dimensione, in byte, di un file

*Ad esempio con **c= FileLen("toto.txt")** viene collocata in c la lunghezza del file *toto**

Ed ora al lavoro!



La mappa riassume tutto il percorso fatto evidenziando gli argomenti trattati in quest’ultima fase: il recupero o dell’invio di dati tra memoria centrale e memorie secondarie e i procedimenti e istruzioni che rendono possibile queste operazioni.

Perché può essere utile procurarsi i dati d’ingresso dal dischetto? _____

perché può essere utile registrare su dischetto i dati in uscita? _____

Quale caratteristica, hanno i file che noi utilizziamo per depositare i dati? _____

Cosa è un campo? _____

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

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