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

Unità di Apprendimento STRUTTURE DI DATI
area delle Informazioni
2 media

1. Strutture di dati nella memoria interna del calcolatore
La necessità di chiamare con un unico nome un insieme organizzato di dati (struttura di dati).
Le variabili semplici e le variabili complesse o **matrici**

1.a - Il **dimensionamento** (DIM), la **dichiarazione**

1.b - Le **matrici**, ad una dimensione (**vettore**), a **due**, a **tre**, ecc….
La cancellazione delle strutture (ERASE)

2. Ingresso e uscita verso le memorie secondarie
La memorizzazione di archivi di dati su **file sequenziali**
Il segnale di "fine del file" (EOF)
In VISUAL BASIC

3. Programmi complessi:
la gestione di una categoria di problemi

3.a - **la programmazione strutturata** e il grafico di progetto
Procedura principale, **procedure**, **procedure comuni**
La gestione con chiamate e ritorni (GOSUB RETURN)
La renumerazione (RENUM) e la fusione di programmi (MERGE)
In VISUAL BASIC

- **premessa**

Volendo utilizzare il calcolatore non solo per la gestione di semplici variabili ma anche di strutture complesse di dati, è importante ricordare che, al contrario della nostra mente, esso non è in grado di collegare le informazioni in base al loro significato. Spetta a noi, quando elaboriamo dei programmi, organizzare i dati in modo logico e funzionale al problema che intendiamo risolvere.
Come già visto nell'U. di A. Algoritmi, il calcolatore ci permette di gestire sia la conoscenza di tipo procedurale che quella dichiarativa. Per farlo organizza la sua memoria RAM in due distinte aree: una serve a memorizzare i programmi, l'altra a memorizzare i dati.

- Il calcolatore accede alla ***memoria programmi*** in modo *sequenziale*; parte leggendo ed eseguendo la prima istruzione e poi tutte le altre, una dopo l'altra. Il programma presente nella memoria del calcolatore mantiene la stessa forma sequenziale quando viene salvato su una *memoria secondaria* come **file** (file BASIC o file eseguibile).
- L'area che serve a ***memorizzare i dati*** ha invece un *accesso diretto.* Ogni spazio di memoria è raggiungibile grazie ad un suo ***indirizzo***. E' il calcolatore che, leggendo nel programma il nome di una variabile e conoscendone l'indirizzo, ne raggiunge il contenuto senza dover passare per altri spazi di memoria. Ma anche queste informazioni, quando vengono memorizzate su ***memoria secondaria***, dovranno assumere la struttura di un file sequenziale formato da una lista di dati memorizzati in **campi**. Il nome del file servirà a definire l'intero archivio; in genere esso ha il suffisso ".txt".

STRUTTURE DI DATI NELLA MEMORIA INTERNA DEL CALCOLATORE

Anche nell'area variabili della ***memoria centrale*** del calcolatore (RAM) è possibile raggruppare sotto un unico nome **strutture di dati** differenti dalla variabile semplice. Queste strutture vengono chiamate variabili complesse o **matrici**. La loro gestione è sostanzialmente diversa da quella che avviene nella nostra mente visto che il calcolatore non è in grado di stabilire legami di significato tra le informazioni che memorizza.

In precedenza abbiamo paragonato le variabili a cassetti dentro i quali è possibile memorizzare dei dati, ora diciamo che è possibile raggruppare blocchi di cassetti sotto un unico nome che verrà stabilito con le stesse modalità viste per le variabili semplici. Il nome indicherà anche se la struttura è destinata ad ospitare dati numerici o alfanumerici. I cassetti appartenenti allo stesso blocco vengono chiamati **elementi** e potranno essere distinti tra di loro grazie a dei numeri progressivi, detti **indici**, che partono sempre dal numero zero.

Se per svolgere un programma è necessario disporre di strutture di dati sarà prima necessario **dimensionarle**, sarà necessario cioè collocare, in genere all'inizio del programma, un istruzione che, dando nome e caratteristiche, ordini al calcolatore di creare la struttura. Se la parte di RAM che il sistema operativo destina ad ospitare i dati parametrici non è sufficiente il calcolatore al dimensionamento darà un segnale di errore. L'ordine è:

➢ **DIM** *nome della struttura* **(***dimensioni della struttura***)**

Dimensionamento / dichiarazione

Il **dimensionamento** è l'unica operazione che può essere fatta sull'intera struttura. Negli esempi che faremo in seguito, per dimensionare le strutture utilizzeremo dati parametrici. Ad esempio *DIM A$(B)* dove il valore di B sarà stato precedentemente chiesto all'operatore.

E' però importante tenere presente che molti moderni linguaggi di programmazione collocano obbligatoriamente il dimensionamento delle strutture all'inizio del programma. Questa fase iniziale, che viene chiamata **dichiarazione**, richiede dei valori costanti che vengono stimati dal programmatore in base alle dimensioni massime previste.

> *Se ad esempio devo dimensionare una struttura che ospiterà i cognomi degli alunni di una classe di scuola media non supererò il numero di 30 elementi, visto che l'attuale legislazione impedisce che si superi tale numero.*

Esaminiamole ora le varie strutture:

- **la matrice ad una dimensione o vettore**

E' la struttura ideale quando dobbiamo memorizzare i valori di un unità informativa. Mettiamo infatti di voler memorizzare tutti i cognomi degli alunni del nostro gruppo di lavoro classe e, successivamente, dato un cognome far scrivere al calcolatore se quel cognome è presente.

Se avessimo a disposizione solo le semplici variabili dovremmo scrivere un programma molto lungo; tanto più lungo quanto più numerosi sono i dati da confrontare.

```
INPUT "Cognome del primo alunno"; A$
INPUT "Cognome del secondo alunno"; B$
INPUT "Cognome del terzo alunno"; C$
        .....
INPUT "Cognome da cercare"; COGN$
IF COGN$=A$ OR COGN$=B$ OR COGN$=C$ OR ..... THEN CLS : PRINT "Presente"
```

E' molto più semplice utilizzare un vettore per depositarvi, nei singoli elementi, i cognomi degli alunni. Si potrà usare:

➢ **DIM** *nome del vettore* **(***numero degli elementi*****)**

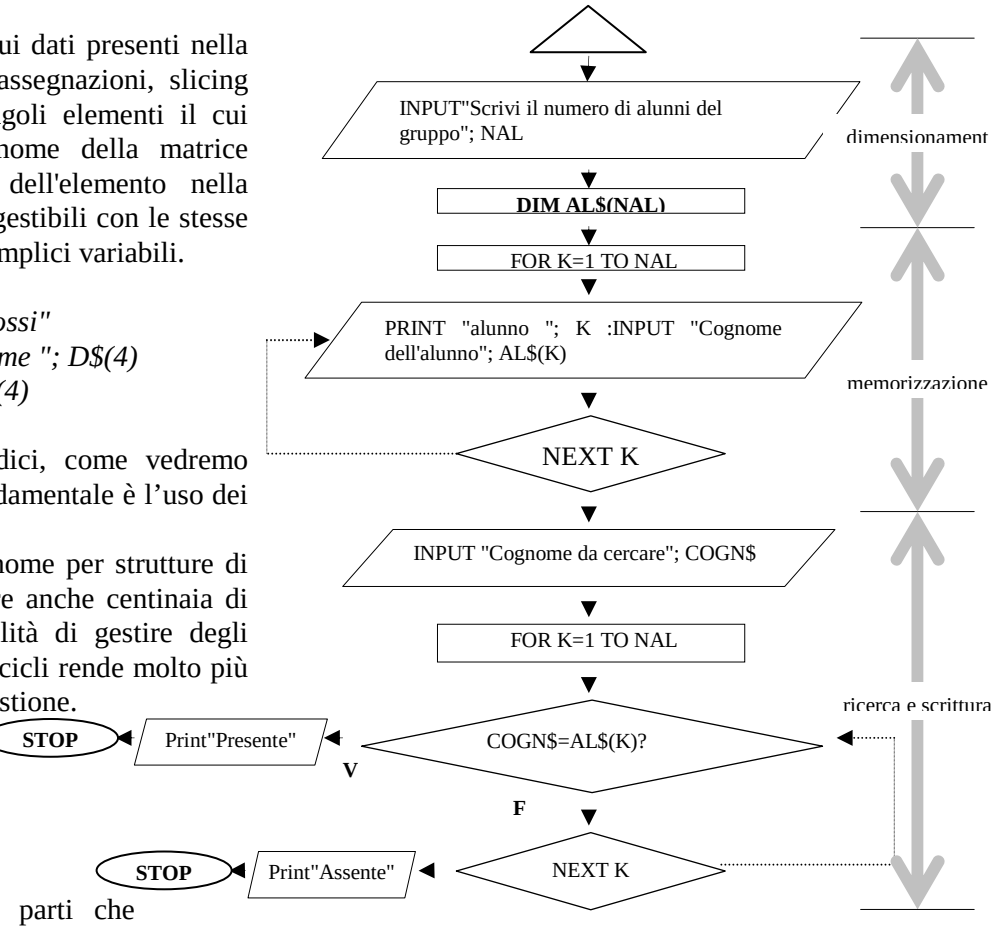
*da far corrispondere al n° degli alunni

Ad esempio: inserendo all'inizio del programma DIM COGN$(22) il calcolatore organizza nella memoria variabili un vettore di 23 elementi numerati da 0 a 22. Questo vettore potrebbe memorizzare 23

cognomi ma è consuetudine ignorare l'elemento zero in modo da far coincidere il primo cognome con l'elemento uno.

Tutte le altre operazioni sui dati presenti nella struttura (assegnazioni, riassegnazioni, slicing ecc..) vanno fatte sui singoli elementi il cui nome è composto dal nome della matrice seguito dalla posizione dell'elemento nella stessa (indice). Essi sono gestibili con le stesse modalità già viste per le semplici variabili.

esempi: A$(1)="Rossi"
INPUT"nome "; D$(4)
PRINT D$(4)



Per la gestione degli indici, come vedremo nell'esempio a fianco, fondamentale è l'uso dei cicli:

La presenza di un unico nome per strutture di dati che possono contenere anche centinaia di informazioni e la possibilità di gestire degli indici parametrici con dei cicli rende molto più semplici i programmi di gestione.

A fianco possiamo vedere risolto il problema visto all'inizio (senza utilizzare nel vettore l'elemento zero).

Sono evidenziate le tre parti che compongono il procedimento.

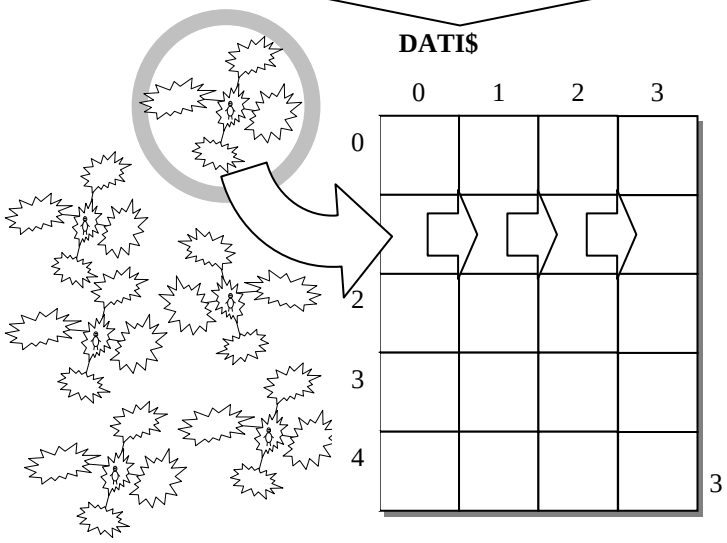
- **la matrice bidimensionale**

Quando ho l'esigenza di memorizzare i valori di un archivio la struttura ideale è la matrice bidimensionale. Su ogni riga potrò collocare un unità informativa mentre sulle colonne saranno allineate le stesse proprietà. In questi casi è necessaria la **regolarità** dell'archivio che viene memorizzato. Ogni unità informativa deve contenere lo stesso numero e tipo di proprietà, che devono essere collocate nello stesso ordine.

Per il dimensionamento si userà l'ordine:

➢ **DIM** *nome della matrice* **(***n° righe***,** *n° colonne***)**

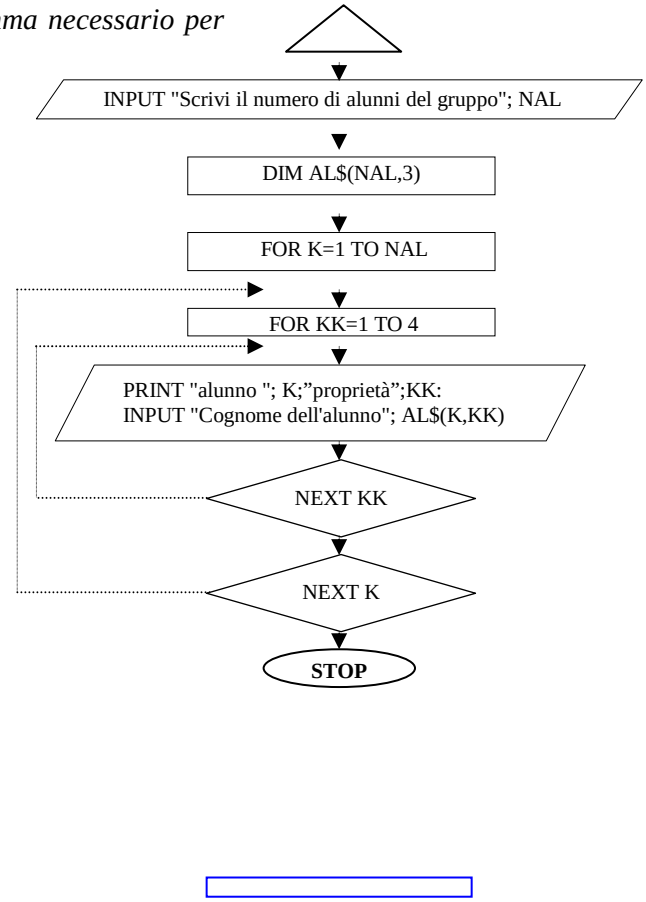
con l'operazione di memorizzazione realizziamo, nella memoria RAM del calcolatore, una copia dei valori dei dati già presenti nella nostra memoria



Se, ad esempio, per ogni componente di un gruppo di lavoro formato da quattro alunni voglio memorizzare tre proprietà (oltre al cognome, il nome e la data di nascita), inserendo all'inizio del programma DIM DATI$(4,3) il calcolatore organizza nella memoria variabili una matrice alfanumerica chiamata DATI$ formata da cinque righe numerate da zero a quattro e quattro colonne numerate da zero a tre. Anche in

questo caso ignorerò riga e colonna zero.

E questo è il diagramma di flusso con il programma necessario per memorizzare i dati:



Nel programma a fianco i numeri contenuti nei contatori K e KK vengono scritti prima dell'INPUT per ricordare all'operatore qual è l'alunno e quali sono le proprietà che si stanno chiedendo.
E' possibile anche scrivere, invece del numero, il nome della proprietà che si sta chiedendo, però in questo caso bisognerà rinunciare al ciclo nidificato.

Scrivi qua sotto il programma così modificato

10	INPUT"Scrivi il numero di alunni del gruppo"; NAL
20	
30	
40	
50	
60	
70	
80	

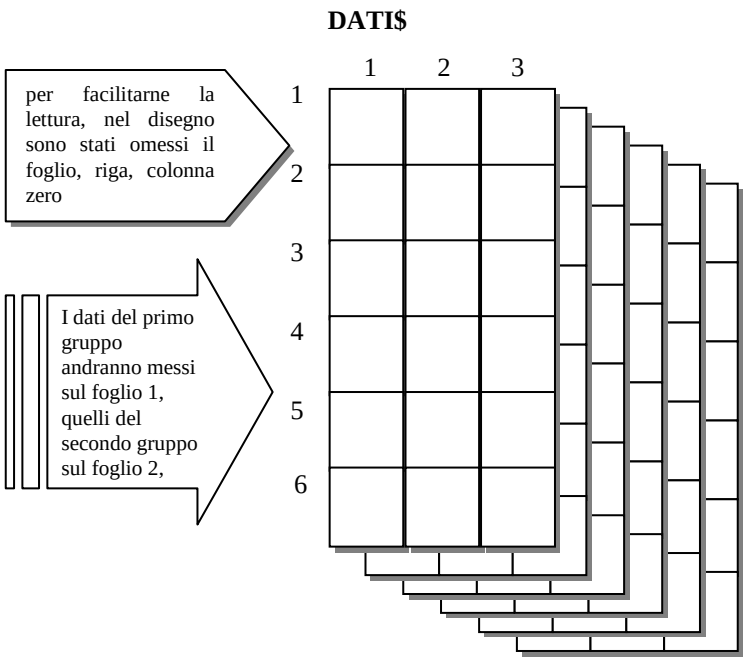
- **matrice tridimensionale**

Immaginiamo di avere l'esigenza di memorizzare le tre proprietà già citate (cognome, nome e data di nascita) anche per gli alunni degli altri sei gruppi di lavoro della nostra classe. Potrò dimensionare una matrice tridimensionale con l'ordine:

➢ **DIM** *nome della matrice* ***(****n°"fogli"*****,** *n° righe*****,** *n° colonne******)**

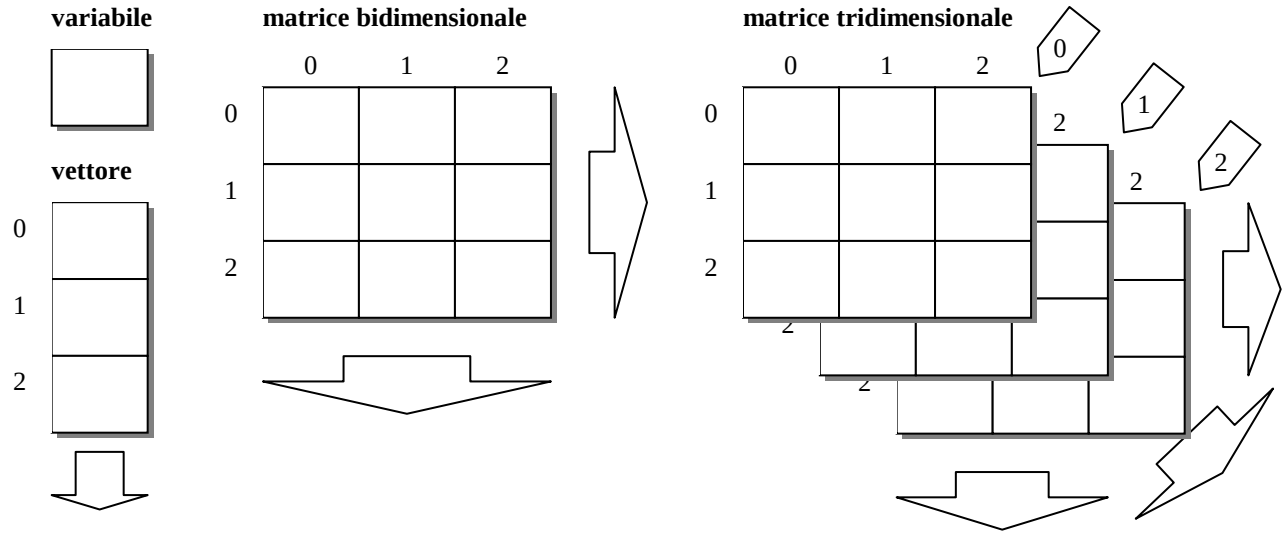
*da far corrispondere al n° dei gruppi
**da far corrispondere al n° degli alunni
***da far corrispondere al n° delle proprietà

Ad esempio: inserendo all'<u>inizio del programma</u> DIM DATI$(6,6,3) il calcolatore organizza nella memoria variabili una matrice alfanumerica chiamata DATI$ formata da sette fogli numerati da 0 a 6, sette righe numerate da 0 a 6 e quattro colonne numerate da 0 a 3.
Anche in questo caso ignoreremo foglio, riga e colonna zero e dunque avremo a



disposizione per memorizzare i dati sei fogli, sei righe, tre colonne.
Per garantire la regolarità dell'archivio abbiamo dimensionato prevedendo sei alunni per gruppo anche se questo numero non viene mai raggiunto.

I seguenti disegni illustrano le caratteristiche delle principali strutture organizzabili nella memoria RAM del calcolatore:



Come si è visto la matrice tridimensionale non è altro che l'unione, sotto lo stesso nome, di matrici bidimensionali. Anche in questo caso fondamentale è la **regolarità**: tutti i fogli devono infatti avere lo stesso numero di righe e di colonne. Questo significa che il numero di righe da usare nel dimensionamento corrisponderà al numero di alunni della classe più numerosa. In mancanza di alunni nei fogli relativi alle altre classi le ultime righe rimarranno vuote.

Cancellazione delle strutture

Con l'ordine:

- **ERASE** *nome della matrice*

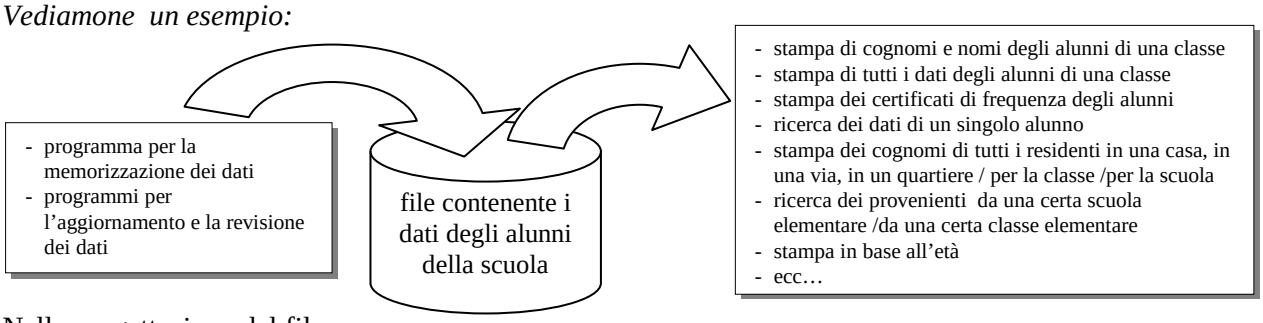
è possibile cancellare dalla memoria RAM una o più strutture di dati recuperando spazio nella RAM e permettendo un loro eventuale nuovo dimensionamento.

In VISUAL BASIC

ritroveremo tutto ciò che abbiamo visto sulle strutture di dati. Cambieranno però alcune regole sul loro utilizzo. Ma di questo ce ne occuperemo l'anno prossimo.

INGRESSO E USCITA VERSO LE MEMORIE SECONDARIE

Come abbiamo visto un file è rappresentato da una lunga fila di campi separati tra di loro dal **segno di fine campo**. Quando il programmatore organizza il trasferimento di un archivio di dati su un file ne dovrà progettare con cura l'organizzazione. Il sistema migliore è quello di farne un disegno su un foglio di carta, disegno che sarà utile quando dovranno essere realizzati i programmi di gestione che prevederanno la lettura dei dati contenuti in quel file. Infatti, una volta che i dati di un archivio sono stati depositati su un file, possono essere numerose le applicazioni che ne possono prevedere l'utilizzo.
Vediamone un esempio:

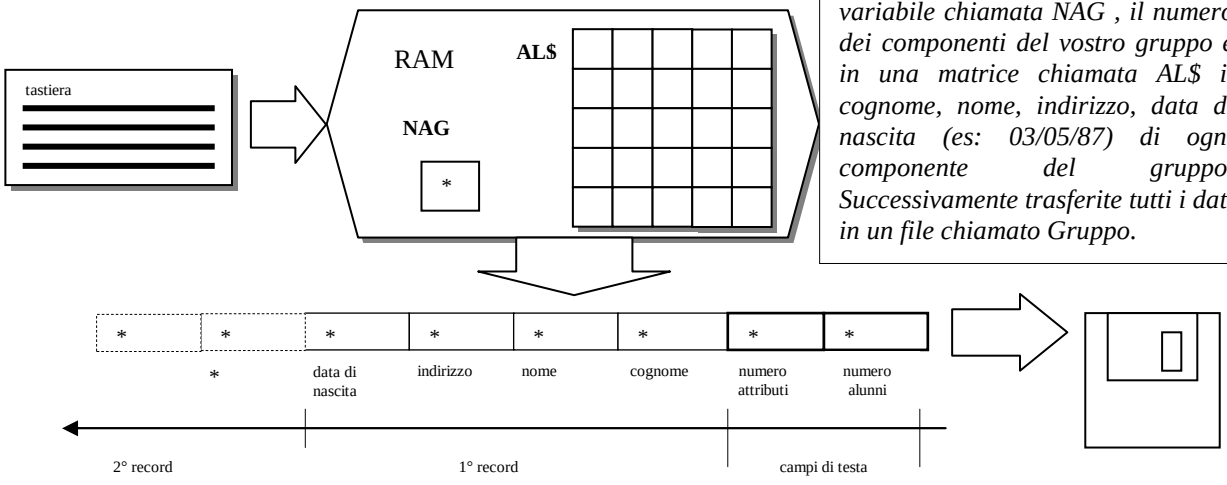


Nella progettazione del file:
- chiameremo **campo** lo spazio di memoria raggiungibile sequenzialmente (in questo caso esso ospita un singolo valore)
- chiameremo **record** la somma degli spazi di memoria (campi) che saranno occupati dai valori di un unità informativa.

E' bene tenere presente che mentre il campo rappresenta un entità realmente esistente sul file, il record è un entità logica che serve al programmatore per gestire in modo ottimale i dati contenuti nel file.
E' anche importante, quanto si progetta un archivio, essere previdenti ed <u>eventualmente raccogliere anche quello informazioni che possono sembrare non immediatamente utili</u>.

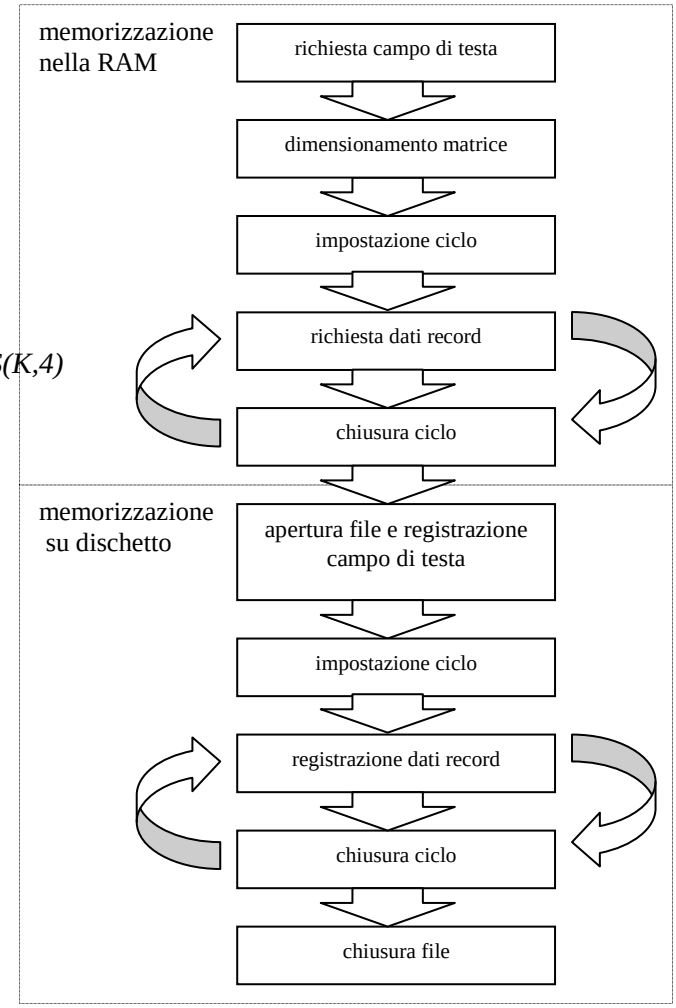
Proprio perché un file depositato su disco deve essere disponibile per diverse utilizzazioni è bene che esso contenga nella sua parte iniziale tutti quei dati che saranno necessari per poterne effettuare la lettura. Un file contenente dei dati sugli alunni di una classe difficilmente potrà essere letto se al suo inizio non sono registrati il numero di alunni della classe (numero di **oggetti**) e il numero di dati registrati per ogni alunno (numero di **proprietà**). I campi iniziali contenenti i dati necessari per la lettura dei record che formano un file vengono chiamati **campi di testa**. Vediamo dunque come progettare il deposito dei dati di un archivio su un file facendo un esempio:

Dovete memorizzare in una variabile chiamata NAG , il numero dei componenti del vostro gruppo e in una matrice chiamata AL$ il cognome, nome, indirizzo, data di nascita (es: 03/05/87) di ogni componente del gruppo. Successivamente trasferite tutti i dati in un file chiamato Gruppo.



Il problema può essere risolto con il seguente programma che viene descritto, nel suo funzionamento, dallo grafo disegnato sul fianco. Analizza bene entrambi.

```
REM .. memorizzazione dei dati sulla RAM
REM
INPUT"Numero alunni del gruppo"; NAG
DIM AL$(NAG,4)
FOR K=1TO NAG:CLS
PRINT "Alunno numero ";K
INPUT "Scrivi il cognome";AL$(K,1)
INPUT "Scrivi il nome"; AL$(K,2)
INPUT "Scrivi l'indirizzo";AL$(K,3)
INPUT "Scrivi la data di nasc.(es: 03/05/87)";AL$(K,4)
NEXT K
REM.. memorizzazione su dischetto
REM
OPEN"O" ,2,"Gruppo"
PRINT#2,NAG;",";"4"
FOR K=1TO NAG
FOR KK=1TO 4
PRINT#2,AL$(K,KK)
NEXT KK
NEXT K
CLOSE 2
STOP
```



Il segnale di "fine del file" (EOF)

Quando viene eseguito l'ordine di chiusura il drive registra in coda al file che si sta chiudendo un apposito segnale che viene chiamato **segno di fine file** (End Of File). In BASIC vi è un apposito ordine studiato per seguire la lettura di un file. Esso è molto utile nel caso non sia possibile conoscere il numero di campi che formano il file. In questo caso l'individuazione del segno di fine file può evitare di proseguire la lettura del file con il conseguente segnale di errore. L'ordine:

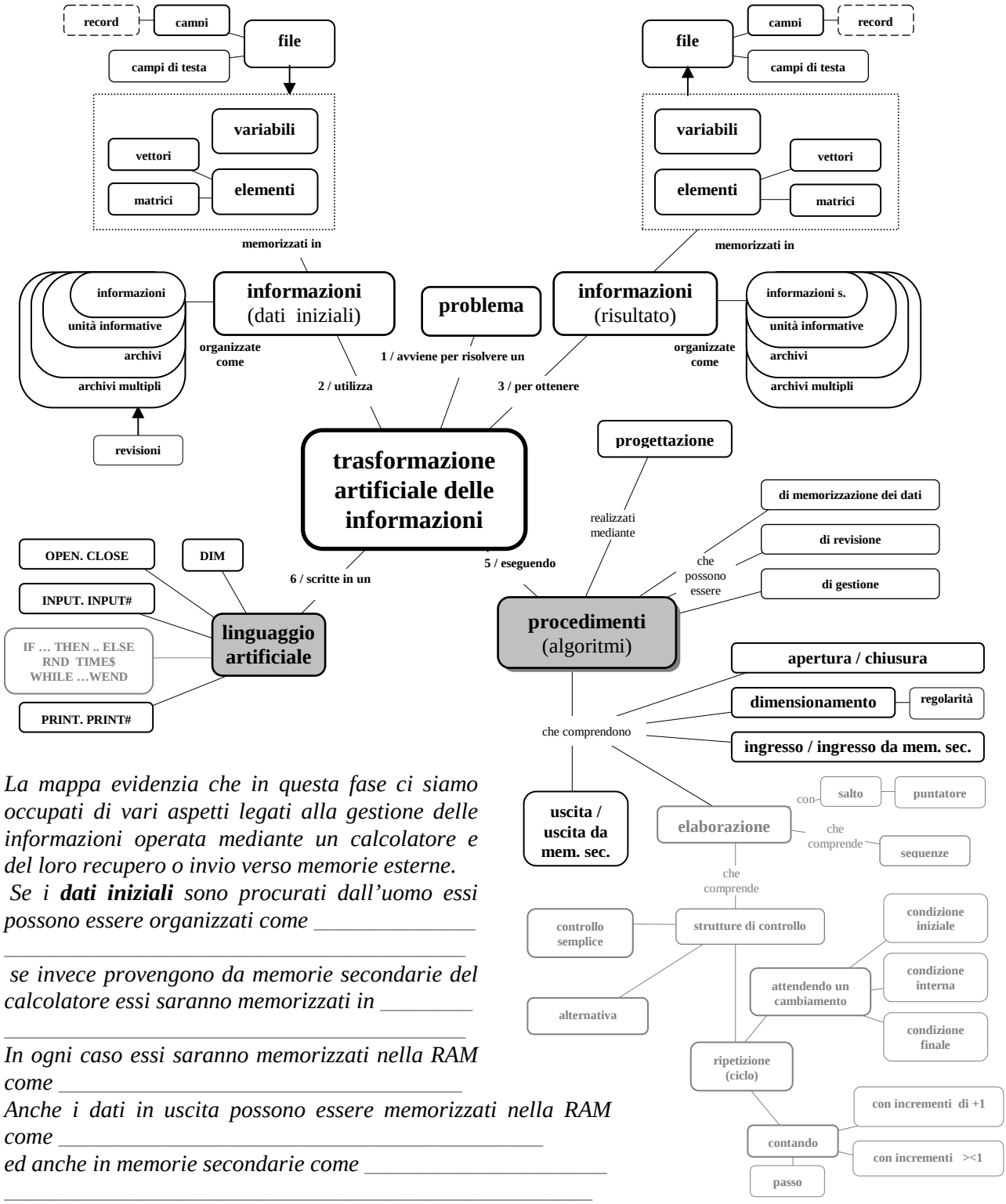
➢ EOF (*n° file*)

conterrà il valore 0 se dopo il campo appena letto vi è il segno di fine campo mentre conterrà il valore –1 se dopo la lettura del campo è stato rilevato il segno di fine del file. In questo caso vuol dire che si è arrivati alla fine del file e che dunque è il momento di chiuderlo.

```
10    OPEN"i",1,"classe"
20    While EOF(1) = 0
30    Input #1, A$
40    Print A$
50    Wend
60    Close1
70    End
```

*Ad esempio il programma a fianco ci permetterà di leggere un file chiamato **classe** che registra i dati di un numero imprecisato di alunni. Con questo programma, però, i dati una volta letti vanno persi. Cosa devo fare se invece li voglio mantenere nella RAM del calcolatore?*

Ed ora al lavoro!



La mappa evidenzia che in questa fase ci siamo occupati di vari aspetti legati alla gestione delle informazioni operata mediante un calcolatore e del loro recupero o invio verso memorie esterne.

*Se i **dati iniziali** sono procurati dall'uomo essi possono essere organizzati come* _____

_____

se invece provengono da memorie secondarie del calcolatore essi saranno memorizzati in _____

_____

In ogni caso essi saranno memorizzati nella RAM come _____

Anche i dati in uscita possono essere memorizzati nella RAM come _____

ed anche in memorie secondarie come _____

_____

*Ci siamo occupati degli aspetti procedurali e in particolare del **dimensionamento** che permette di*

_____

e delle istruzioni che consentono tale operazione come _____

*E dell'**apertura** e **chiusura** che permette di* _____

utilizzando gli ordini _____

*Le strutture di dati archiviate nel calcolatore devono essere **regolari**; significa che* _____

_____

PROGRAMMI COMPLESSI

La gestione di una classe (o categoria) di problemi

Buona parte dei programmi "professionali" che vediamo in azione sui calcolatori di banche, uffici pubblici, aziende private ecc... entrano in funzione con l'accensione dei calcolatori e terminano il loro funzionamento con il termine delle attività lavorative. Durante tutta la giornata l'operatore sceglierà, all'interno di un "menù" che compare sullo schermo, il tipo di attività che il calcolatore dovrà svolgere. Al termine di questa attività il calcolatore tornerà al menù principale pronto a svolgere nuovi lavori. Questo evita all'operatore, spesso inesperto, di dover caricare di volta in volta il programma necessario.

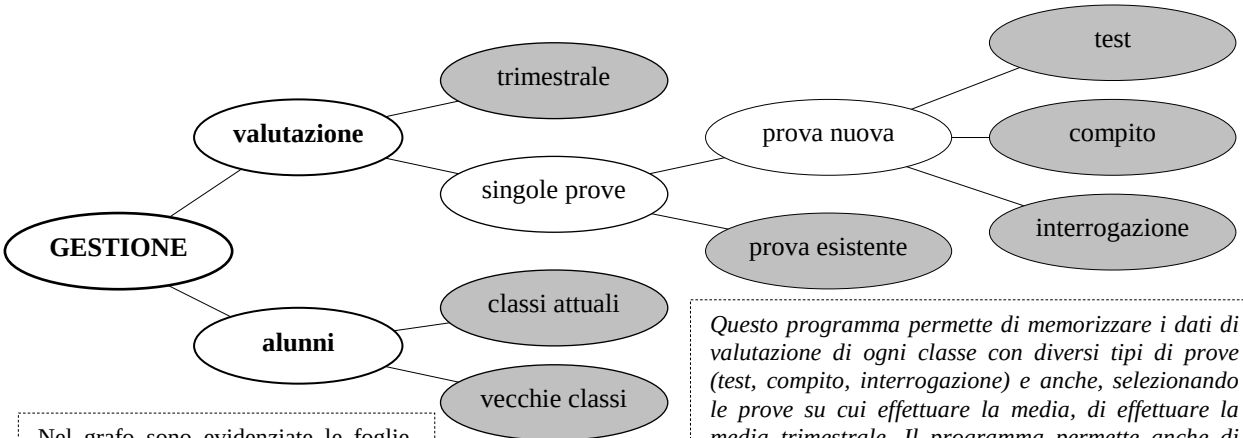
Per lavorare in questo modo ("user frienly") è però necessario realizzare grandi programmi che contengono tutto ciò che il calcolatore può svolgere in quel campo di attività.

Chi progetta questi programmi (abbandonati gli oramai non più utili diagrammi di flusso) dovrà prendere in esame l'ambiente di lavoro nel quale essi verranno utilizzati ed individuare tutte le attività che fanno capo a quell'ambiente (ad esempio tutte le attività che possono essere svolte allo sportello di una banca oppure a quello di un ufficio anagrafe).

Individuare i collegamenti tra le attività connesse ad una determinata situazione di lavoro è fondamentale per poi realizzarne un buon programma di gestione. Per realizzare tale attività di studio può essere utile utilizzare dei grafi di scomposizione (vedi U.D. Algoritmi) questa volta non per collocarvi le singoli azioni da compiere, ma per individuare i problemi semplici in cui suddividere il problema complesso.

La programmazione strutturata

Il **grafico di progetto** è dunque un albero di scomposizione utilizzato per individuare le parti in cui dividere il campo di attività preso in esame. Vediamo ora l'albero di scomposizione realizzato per la progettazione di un programma chiamato "gestione" con cui il nostro insegnante di E.T. gestisce la valutazione degli alunni. Attraverso vari livelli di scomposizione possiamo individuare i diversi problemi in cui può essere scomposto, alcuni dei quali sono a loro volta scomponibili in problemi più semplici:



Nel grafo sono evidenziate le foglie, cioè i diversi problemi che è possibile risolvere; a cui si accede dopo una serie di scelte

Questo programma permette di memorizzare i dati di valutazione di ogni classe con diversi tipi di prove (test, compito, interrogazione) e anche, selezionando le prove su cui effettuare la media, di effettuare la media trimestrale. Il programma permette anche di ottenere, su schermo e su stampante, gli elenchi degli alunni delle classi attuali e di quelle che oramai hanno lasciato la scuola

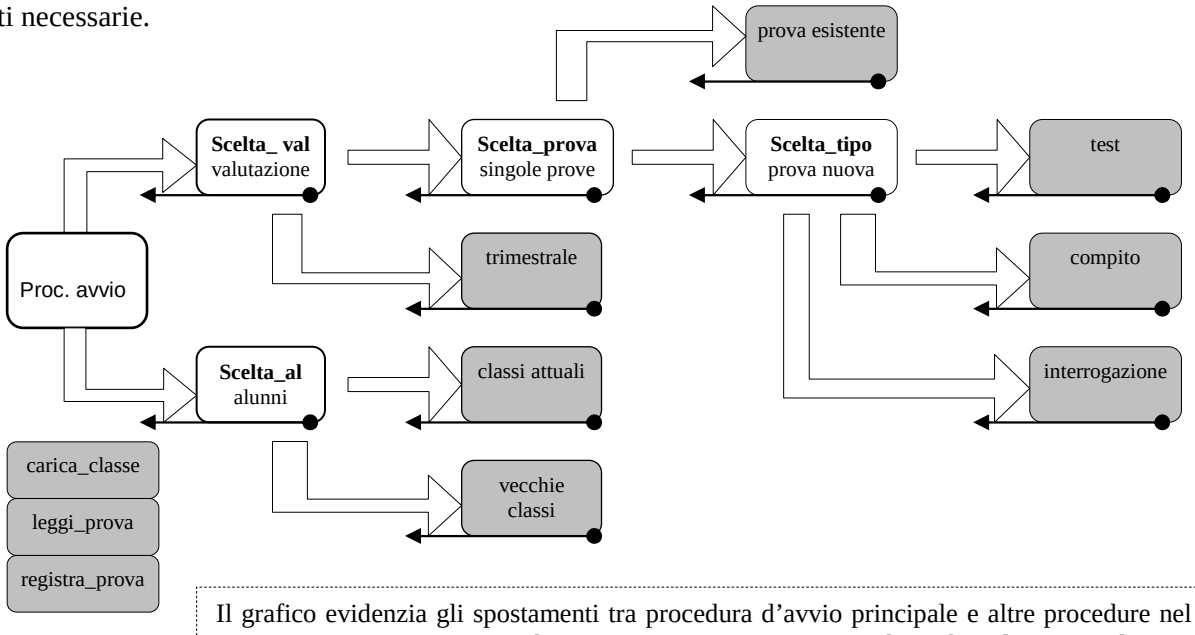
Nella nostra realtà quotidiana noi siamo abituati a suddividere un problema complesso in problemi più semplici, a ognuno dei quali assegniamo un ***nome*** che ha un ***legame di significato*** con il ***problema semplice*** che intende risolvere (*ad es. chiameremo "apparecchiare la tavola" la serie di azioni semplici che compiamo ogni volta che dobbiamo preparare la tavola per mangiare*).

Quando si deve realizzare un programma per gestire un problema complesso è utile lavorare con le stesse modalità. E' bene infatti dare un nome alle singole procedure che formano il programma. Questa possibilità permette di organizzare la propria attività di programmazione secondo un metodo che normalmente viene chiamato ***programmazione strutturata***.

Procedura principale, procedure, procedure comuni

Vi sarà una **procedura principale** (o d'**avvio**) nella quale saranno collocate una serie di istruzioni di chiamata (dette **richiami di procedura**) delle procedure in cui è stato scomposto il problema complesso. La sua funzione è solo quella di indicare al calcolatore, in sequenza, i nomi delle procedure in cui è stato scomposto il problema e che saranno da eseguire quando sarà dato l'ordine di esecuzione. La procedura d'avvio è spesso chiamata **menù principale** perché la chiamata delle procedure è collegata a controlli effettuati su dati immessi dall'operatore. Anche nell'esempio appena fatto, dopo il **menù principale** vi sono altri **menù subordinati** (che nel grafo di scomposizione corrispondono ai nodi) che permettono di effettuare in successione la serie di scelte che porta alla procedura che deve essere eseguita.

E' anche compito della procedura d'avvio contenere alcuni ordini di carattere generale che servono per preparare il calcolatore al lavoro come, ad esempio, il **dimensionamento/dichiarazione** delle strutture di dati necessarie.



Il grafico evidenzia gli spostamenti tra procedura d'avvio principale e altre procedure nel programma presentato precedentemente. E' importante ricordare che, al termine di ogni procedura, l'esecuzione ritorna alla procedura di provenienza. A parte sono collocate alcune *procedure comuni* destinate ad essere chiamate in più occasioni.

La gestione con chiamate e ritorni (GOSUB …RETURN)

Lavorando in GWBASIC non è possibile dare nomi alle diverse procedure ed è necessario lavorare su un unico "foglio" in cui tutte le istruzioni sono collocate in sequenza precedute dal numero di linea. E' comunque possibile utilizzare il metodo della programmazione strutturata per individuare la procedura d'avvio, che sarà collocato all'inizio e che terminerà con END e le varie procedure che saranno collocate di seguito e che termineranno sempre con un istruzione che permetterà il **ritorno** dell'esecuzione alla procedura di provenienza. La chiamata, durante l'esecuzione, dalla procedura d'avvio o, anche, da un'altra procedura (**nidificazione di procedure**), avviene con una speciale istruzione di salto che utilizza il numero della linea d'inizio della procedura.

E' bene che l'ordine di rinvio sia seguito dall'indicazione della procedura verso la quale esso avviene, preceduto da REM, come è anche bene iniziare una procedura con un REM seguito dal nome e dalla eventuale funzione della procedura che sta iniziando.

Al momento della progettazione del programma è molto importante inserire sul grafico del progetto, accanto ai nomi delle procedure, i numeri delle linee a partire dalle quali prevediamo di collocarle.

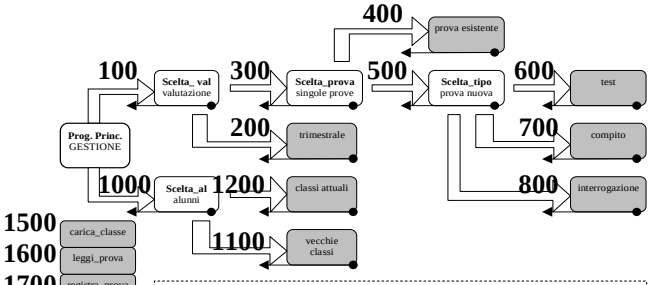


grafico del progetto con una prima ipotesi di collocazione delle procedure. Se necessario, l'ordine RENUM permetterà di modificare i numeri di linea

Per effettuare la chiamata useremo l'ordine:

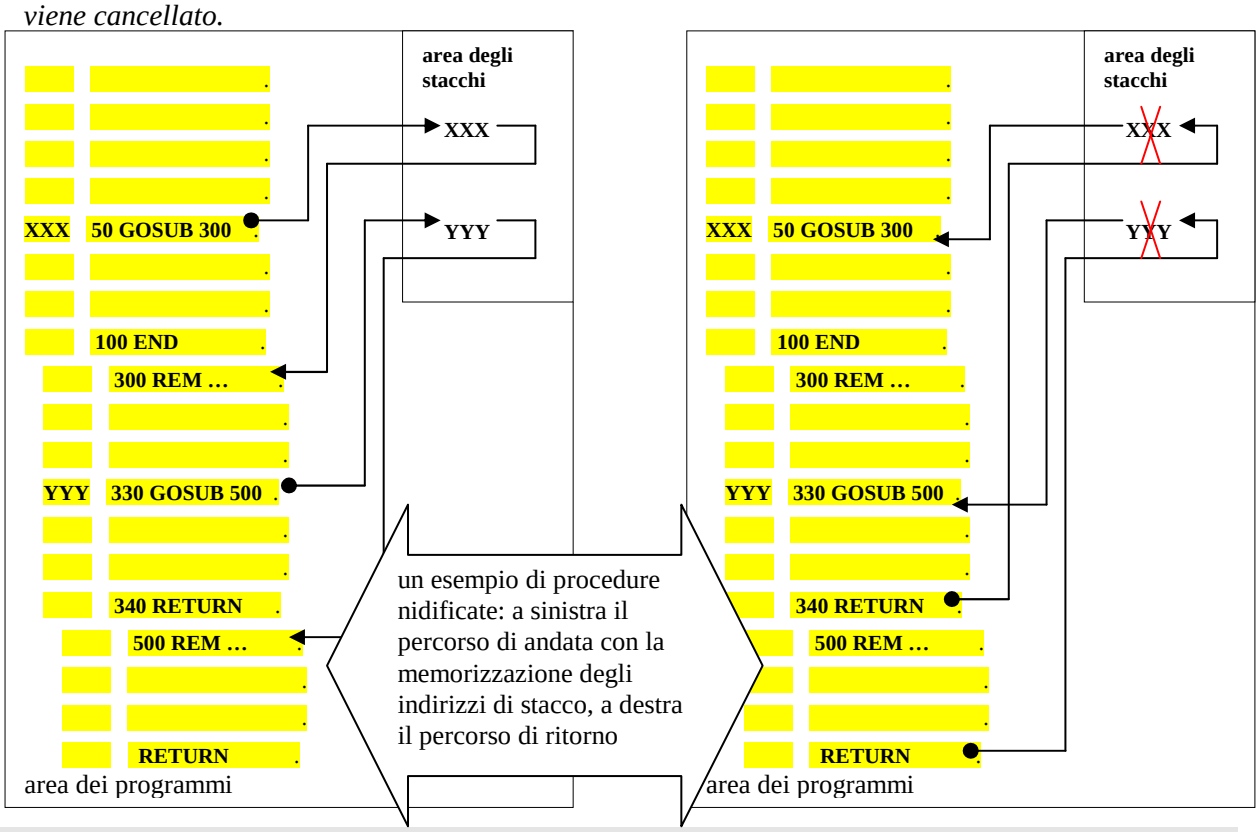
➢ GOSUB *n° linea di destinazione*

Eseguendo questa istruzione il calcolatore salta alla linea indicata, utilizzando l'area degli stacchi per memorizzare l'indirizzo di memoria della linea sulla quale il GOSUB è collocato (linea di partenza o di stacco).
Eseguendo GOSUB il calcolatore raggiunge la linea d'inizio della procedura, sulla quale è bene collocare il nome preceduto da un REM. In coda alla procedura andrà collocato l'ordine:

➢ RETURN

che, eseguito dal calcolatore, farà tornare l'esecuzione del programma all'istruzione successiva all'ultimo GOSUB eseguito.

*Quando, ad esempio, il calcolatore esegue l'istruzione **80 GOSUB 200** il calcolatore, dopo aver memorizzato l'indirizzo di memoria della linea 80 (linea di stacco), raggiunge la linea 200 ed esegue le linee seguenti. In coda alla procedura l'ordine RETURN manda il calcolatore a leggere l'indirizzo dell'ultima linea di stacco, nell'area degli stacchi, e vi tornerà riprendendo l'esecuzione del programma dall'istruzione successiva a quella di chiamata. Una volta letto, l'indirizzo di stacco viene cancellato.*



Nota Bene: Gosub e Return sono validi, anche se poco utilizzati) anche in Visual Basic. Al posto del numero di linea viene utilizzato una stringa di caratteri (anche un numero) seguito da due punti. Molto più utilizzato è il richiamo di procedura con l'ordine **Call**, che vedremo in seguito.

In GWBASIC potrà essere utile nel lavoro di programmazione l'esecuzione in diretta dell'ordine:

➢ RENUM *numero nuovo, numero vecchio, incremento*

Ad esempio con RENUM 1000, 100, 20 il calcolatore modificherà i numeri di tutte le linee dell'attuale programma a partire dalla 100. La numero 100 diventerà 1000 e tutte le successive saranno rinumerate con un incremento di 50. L'esecuzione del solo RENUM rinumera tutto il programma con un incremento di 10.

RENUM si occupa anche di modificare gli ordini di salto (GOTO e GOSUB) rendendoli coerenti con la nuova numerazione.

Se sul lavoro complesso lavorano più gruppi ognuno dei quali ne svolge una parte, potrà essere necessario "incollare" al programma attualmente in memoria un altro precedentemente salvato su dischetto. Per questo si userà l'ordine GWBASIC:

➢ **MERGE** *nome del programma da incollare*

Ad esempio con MERGE "alunni" il calcolatore incolla il programma alunni al programma attualmente in memoria. In caso di numeri di linea identici il calcolatore cancella quelli preesistenti.
ATTENZIONE: con MERGE si possono incollare solo programmi precedentemente salvati su disco in formato ASCII (ad esempio SAVE"alunni",a)

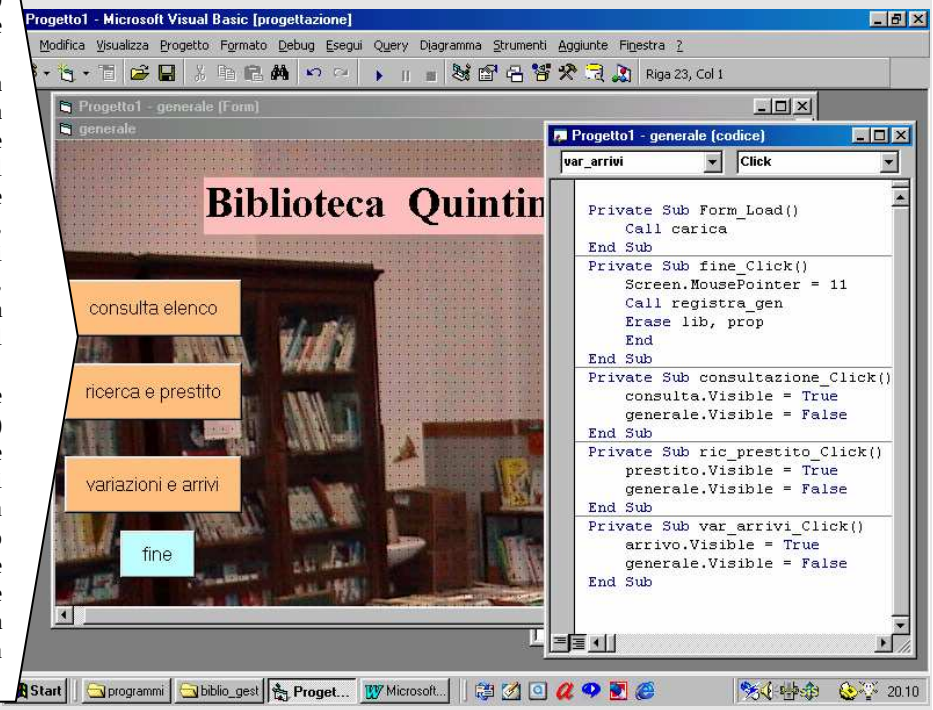
In VISUAL BASIC

espansioni del linguaggio

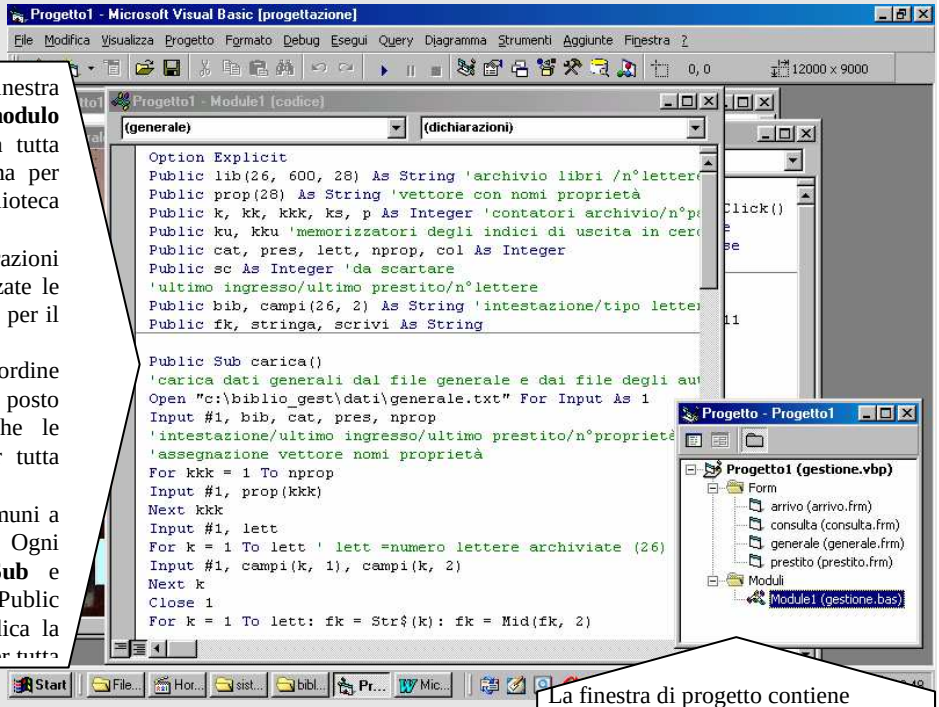
VB è un linguaggio studiato per realizzare applicazioni complesse. Vi è sempre una schermata principale, chiamata **main form**, e che può servire anche da menù per accedere ad eventuali altre schermate e che comprende anche tutto il software necessario per consentire all'utente di operare le scelte, di attivare procedure di calcolo, di immettere e ricevere dati. Un'apposita finestra permetterà di comporre, in fase di progettazione, il disegno della schermata (**form**), mentre un'altra ospiterà le istruzioni (**codice**) che verranno scritte in un modo molto simile alla videoscrittura.

Il form principale (main form) del programma per la gestione della biblioteca scolastica.

- nella finestra di sinistra viene composta la schermata (**form**) che comparirà all'avvio del programma. E' possibile collocare immagini, grafici, testi e pulsanti di opzione che, selezionati, porteranno alle altre form che compongono il programma.
- a destra vi è la finestra che ospita il **codice** (istruzioni) necessario del la gestione del form. Esso, come si può notare è scomposto in procedure che contengono ciò che il calcolatore deve fare in risposta a ciò che l'utente fa sul form (**evento**). Ad esempio un click su un bottone.



Il software per gestire un form e il suo codice viene salvato a parte, in un file con estensione **.frm** (che può essere affiancato da un altro file con estensione **.frx**). Un applicazione VB può dunque essere formata da vari file .frm ognuno dei quali contiene il disegno del form e il relativo codice.
Questa organizzazione del software permetterà al calcolatore di non caricare in RAM tutto il programma ma solo quello o quei file che sono necessari per l'attività che si intende svolgere.
Quella parte del software (dimensionamento ed organizzazione delle strutture di dati, procedure di interesse generale) che non è legato ad un singolo form verrà composto in un'apposita finestra (chiamata **modulo**) e salvata in un file che avrà un'estensione **.bas**.



In primo piano si vede la finestra contenente il codice del **modulo** (cioè il codice che interessa tutta l'applicazione) del programma per la gestione della biblioteca scolastica.

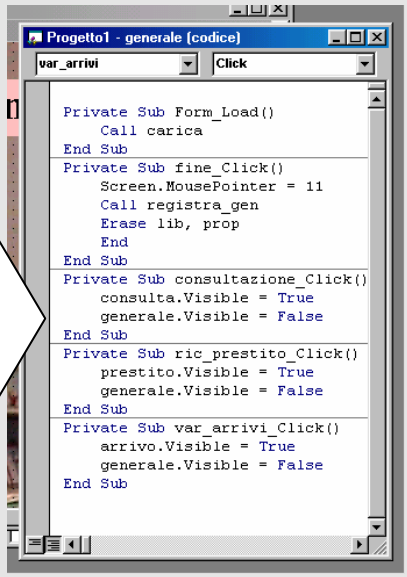
- In alto vi sono le dichiarazioni con cui vengono organizzate le strutture di dati necessarie per il funzionamento dell'applicazione. L'ordine **Public** viene utilizzato al posto di Dim per indicare che le strutture sono valide per tutta l'applicazione.
- Seguono le procedure comuni a tutta l'applicazione. Ogni procedura inizia con **Sub** e finisce con **End Sub**. Public anche in questo caso indica la validità della procedura per tutta

La finestra di progetto contiene l'elenco dei form componenti l'applicazione e indica la presenza di eventuali moduli.
Può essere utilizzata per aprire le varie finestre che compongono l'applicazione.

Il **file di progetto** (estensione **.vbp**) ha il compito di memorizzare la mappa dei file che formano un'applicazione. Ha dunque il compito di avviare l'applicazione attivando poi i file che saranno necessari.

le procedure

In VB una procedura è sempre compresa tra:
Sub *nome della procedura*
istruzioni
End Sub
L'ordine Sub può essere preceduto da Public o Private a seconda se la procedura è valida per tutta l'applicazione o per un solo form.
Per agevolare il programmatore, come vedremo, VB provvede automaticamente a delimitare la procedura e spesso vi da anche un nome provvisorio. Queste scelte possono comunque essere modificate dal programmatore.
Per richiamare una procedura viene utilizzato l'ordine:
Call *nome della procedura*



Rivediamo la finestra di codice del main form:

- la procedura **Form** che si attiva al caricamento (**Load**) richiama la procedura **carica** che abbiamo già visto nel modulo
- la procedura **fine** che si attiva cliccando (**Click**) sul pulsante "fine" del form richiama la procedura **registra_gen**, anch'essa nel modulo, dopo aver trasformato la freccia del mouse in una clessidra (screen.MousePointer=11). Dopo che Erase ha cancellato le strutture di dati dalla RAM, l'ordine End pone fine all'esecuzione del programma
- le altre procedure, attivate cliccando i rispettivi pulsanti, si limitano a rendere non visibile il form principale (generale) e a rendere visibili e attivi gli altri form e il loro codice

Ed ora al lavoro!

Ora abbiamo le conoscenze necessarie realizzare programmi di gestione per archivi di dati; questi però dovranno essere prima memorizzati su file. E' possibile evitare di costruire programmi di memorizzazione e usare un sistema più pratico e veloce. Dovremo utilizzare il software di videoscrittura Blocco Note, in genere presente su tutti i calcolatori, anche i più vecchi. Dopo aver stabilito oggetti e proprietà che devono essere presenti nell'archivio e chiariti i vari problemi di formato, si inizieranno a scrivere i dati, a partire dai campi di testa.
Quando salveremo, il calcolatore aggiungerà al nome da noi scelto l'estensione .txt per indicare che il file è in formato testo. Ad esempio se salviamo con il nome "dati" dovremo ricordarci di inserire nell'ordine di apertura il nome "dati.txt".

A fianco i dati di una piccola classe di 11 alunni battuti su una pagina di Blocco Note e pronti per essere salvati. I due campi di testa indicano il numero degli alunni (n° dei record) e il numero delle proprietà. Al momento della lettura del file, l'invio a capo viene considerato segno di separazione di campo (come la virgola). Nell'esempio si è preferito usarlo per separare tra loro i vari record mentre la virgola viene usata per separare i campi all'interno dello stesso record. E' importante notare che tra un dato e l'altro vi è solo la virgola e non vi sono spazi bianchi.

```
11,9
Beltrami,Aldo,m,2542678,17/10/87,Cologno Monzese,via Vespucci 4,Milan,nuoto
Brambilla,Alessia,f,27413175,21/09/87,Milano,via Porpora 100,Inter,tennis
Bianchi,Federico,m,28417606,06/02/88,Milano,via Carpi 19,Inter,calcio
Bondani,Laura,f,2446810,05/06/87,Milano,via Adelchi 4,-,ginnastica
Criscuolo,Marco,m,23441920,07/04/87,Milano,via Orione 6,Juventus,calcio
Damiani,Matteo,m,28421396,06/07/87,Vimodrone,via Roma 12,Milan,calcio
Greco,Elisabetta,f,2236541,19/09/87,Segrate,via Amendola 17,Milan,pallavolo
Paravia,Valeria,f,2353458,08/10/87,Milano,via Casoretto 2,Fiorentina,equitazione
Rossi,Edoardo,m,28924877,29/10/87,Segrate,via Olgetta 15,Juventus,calcio
Spadari,Gianni,m,2258537,22/05/87,Milano,via Poggiolini 4,Atalanta,tennis
Vigliani,Andrea,m,2258450,12/07/87,Segrate,via Amendola 6,Juventus,nuoto
```

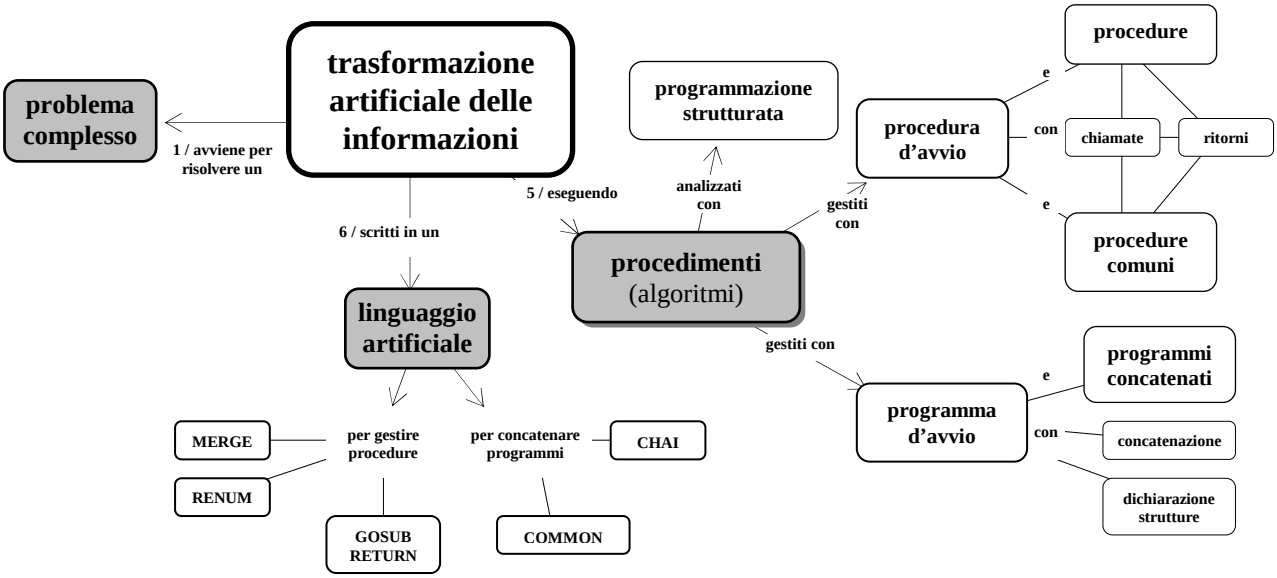
aggiungi l'argomento di altri possibili programmi di gestione realizzabili sull'archivio in aggiunta a quello già scritto

osserva il precedente archivio e segna nella tabella a fianco i nomi delle nove proprietà presenti

1	Dato il cognome di un alunno scrivere tutti i dati dell'alunno

1	cognome dell'alunno
2	
3	
4	
5	
6	
7	
8	
9	

Possiamo ora iniziare a progettare un archivio sulla nostra classe; più sarà ricco di dati, più numerosi saranno i programmi di gestione che potremo realizzare su di esso. Una volta memorizzato il file realizzeremo il programma di lettura e i vari programmi di gestione.



Il problema complesso ci costringe a rivedere la mappa su cui abbiamo lavorato sin dall'inizio. Anche i procedimenti diventano più complessi e richiedono nuovi metodi di lavoro. Verbalizziamo qui di seguito ciò che abbiamo visto sulla mappa.

La trasformazione artificiale delle informazioni avviene _____