UN NUOVO MODO DI PROGRAMMARE

I calcolatori sono in grado di comprendere solo un linguaggio composto da sequenze di uno e di zero. Un linguaggio cioè che si basa sul codice binario e che è noto anche come ***codice macchina***. Esso viene classificato come linguaggio di programmazione di **basso livello**.

Il termine basso /medio /alto riferito ai linguaggi di programmazione non indica la complessità del linguaggio ma la sua "vicinanza" al codice macchina o al linguaggio dell'uomo. Mentre la "logica" del linguaggio di basso livello è più vicina al modo di funzionare del calcolatore quella del linguaggio ad alto livello è più vicina ai linguaggi umani ed in particolare alla lingua inglese.

Tuttavia, scrivere un programma composto da una serie di uno e zero può essere un'impresa difficile, onerosa in termini di tempo e noiosa. Per rendere la programmazione un po' più semplice, gli scienziati inventarono prima il *linguaggio assembly* (ora in pratica poco utilizzato perché troppo complesso) e poi nuovi linguaggi a partire da *A, B, C,* fino a **C++** (tuttora molto usato), tutti considerati linguaggi di **medio livello**. I linguaggi di medio livello tendono a fondere normali parole in inglese e strani simboli (come ad esempio il comando a++). I programmi in C++ sono molto più semplici da scrivere dei programmi in codice macchina o in assembly.

In ogni caso, prima di poter eseguire un programma in C++, il vostro calcolatore dovrà tradurre il programma scritto in C++ in un programma equivalente scritto in codice macchina. Dato che C++ è tanto diverso dal codice macchina, la traduzione dei codice scritto in C++ nel codice macchina richiederà un po' di tempo e di lavoro da parte del calcolatore ma ridurrà la fatica del programmatore.

Infine, i linguaggi di **alto livello**, come il COBOL, il Pascal e il **BASIC**, si servono di parole inglesi e acronimi facilmente leggibili e utilizzabili da tutti. Ovviamente i calcolatori non comprendono i linguaggi di alto livello e dunque prima di poter eseguire un programma scritto in un linguaggio di alto livello, il calcolatore dovrà tradurlo in un programma equivalente scritto in codice macchina.

Dato che l'inglese utilizza le lettere mentre i calcolatori comprendono solo i simboli uno e zero, la conversione di un linguaggio ad alto livello nella serie di uno e zero che costituisce il linguaggio macchina richiede più tempo e più lavoro da parte del calcolatore, ma programmare con uno di questi linguaggi può essere davvero alla portata di tutti.

Sebbene i programmatori spesso sostengono i vantaggi di un linguaggio rispetto a un altro, non esiste un linguaggio migliore da utilizzare per tutte le esigenze di programmazione. E' possibile utilizzare qualsiasi linguaggio per scrivere praticamente qualsiasi tipo di programma, solo che alcuni linguaggi si adattano meglio di altri a determinati scopi.

Se, esclusivamente per motivi di velocità, molti programmatori compiono grandi sforzi per apprendere il linguaggio assembly o il C++ noi, più semplicemente, potremo far fare al calcolatore le stesse cose utilizzando il più semplice VISUAL BASIC.

Rapida storia del linguaggio BASIC

Nell'ormai lontano 1964, programmare un calcolatore era un lavoro noioso e dispendioso in termini di tempo. Nel tentativo di rendere la programmazione più semplice o apprezzabile, due professori di Darimouth (USA), John G. Kemeny e Thomas E Kudz, svilupparono il linguaggio BASIC.

Il BASIC (un acronimo di Beginner's All-Purpose Symbolic lnstruction Code / Codifica a istruzioni simboliche, per principianti, per applicazioni di carattere generale) è stato sviluppato per rendere più accessibile la programmazione del calcolatore. Contrariamente ai linguaggi disponibili a quei tempi, il BASIC era **interattivo**. In pratica, nel momento dell'esecuzione, il calcolatore obbediva oppure segnalava che era stato compiuto un errore su una singola istruzione evitando di proseguire nell'elaborazione.

Il fatto che questo linguaggio segnalasse in tempo reale che era stato compiuto un errore costituì un'importante rivoluzione nella programmazione. Altri linguaggi costringevano l'utente a scrivere tutto il programma, e se venivano immessi degli errori il calcolatore non segnalava nulla prima della fine. Dopodiché, quando il programma non funzionava, si doveva partire alla caccia dell'errore.

Dato che BASIC si rivelò tanto semplice da utilizzare, il linguaggio cominciò a essere offerto da quasi tutti i calcolatori. Assieme al primi PC IBM venne offerta una versione chiamata BASICA, e ben presto i calcolatore IBM compatibili offrirono una versione simile chiamata GW-BASIC.

Con l'uscita di MS-DOS 5.0 fu introdotta anche una nuova versione di BASIC chiamata QBASIC. Tuttavia, dato che in origine il BASIC era stato sviluppato per i principianti, i programmatori seri continuarono a considerarlo come un linguaggio giocattolo.
Tutto cambiò quando Microsoft presentò Microsoft Windows. Non solo i programmatori dovevano scrivere programmi che funzionassero, ma veniva loro richiesto anche di scrivere programmi che offrissero menu a discesa, finestre di dialogo e barre di scorrimento. In pratica, la programmazione diventò doppiamente difficile a causa del complesso ambiente operativo di Windows.
Nel 1991 Microsoft mise a punto Visual Basic, e finalmente i programmatori ebbero la possibilità di cominciare subito a stabilire l'aspetto del loro programma, per potersi poi concentrare sul suo funzionamento. Dato che stabilire l'aspetto di un programma significa compiere già il 50% del lavoro, Visual Basic rende questa parte il più semplice possibile per poi consentirvi di concentrare gli sforzi sulle operazioni che inducono il programma a fare qualcosa di utile.
Grazie a queste caratteristiche, Visual Basic è molto più semplice da imparare rispetto a Pascal o C++. Non solo consente di scrivere i programmi con maggiore facilità, ma richiede anche uno sforzo minore per crearli.

Una breve storia delle interfacce utente e dei sistemi operativi

Ai primordi dei calcolatore (nei lontani anni '50) fare in modo che un calcolatore facesse qualcosa significava aprirlo e riposizionare alcuni cavi.

Negli anni'60 per utilizzare un calcolatore si utilizzavano pile di schede perforate; quando il calcolatore rispondeva ai comandi stampava qualcosa su un pezzo di carta, e questa attività poteva richiedere ore o persino giorni. Di conseguenza, usare un calcolatore era ancora un'attività lenta e noiosa.

Negli anni '70 collegarono una tastiera a un televisore e definirono questo insieme "terminale". Per la prima volta fu possibile digitare un comando direttamente sul calcolatore e ottenere una risposta immediata. Questa capacità avrebbe dovuto rendere i calcolatori più semplici da utilizzare.

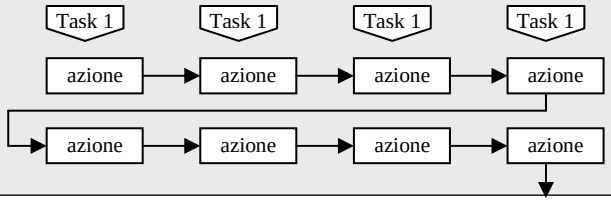
- **interfaccia a riga di comando**
  Però queste prime interfaccia utente, piuttosto rozze, prevedevano solo uno schermo vuoto e un puntino lampeggiante, chiamato cursore. Perché il calcolatore facesse qualcosa era necessario conoscere i comandi corretti da digitare; i programmi che utilizzavano queste brutte interfaccia venivano chiamati ***interfaccia a riga di comando***. Purtroppo, se non si conoscevano i comandi corretti da digitare o i tasti giusti da premere, il calcolatore si rifiutava di collaborare (un esempio di interfaccia a riga di comando è la schermata MS DOS che abbiamo sul monitor prima di caricare il GWBASIC).
  Per rendere i calcolatore più semplici da utilizzare, i programmi cominciarono e elencare i comandi disponibili presentando dei menu sullo schermo: invece di costringere gli utenti a digitare i comandi giusti (e scriverti anche nella maniera corretta), i menu consentirono a chiunque dì scegliere il comando desiderato selezionandolo con il **mouse**.

- **interfaccia grafica utente**
  Ben presto tutti i programmi cominciarono a offrire dei menu; purtroppo tutti i menu funzionavano in maniera diversa, per cui saper utilizzate un programma non significava necessariamente saperne utilizzare anche un altro. Di solito i menu non si assomigliavano in nulla. Per convincere gli, utenti che i calcolatori erano amici, i programmatori definirono un metodo standard per tutti i programmi. Invece di visualizzare sullo schermo simboli e lettere (come C: \DOS>) questo nuovo standard si servì di colori, menu e immagini grafiche. Venne così offerta la possibilità di selezionare i comandi puntando a un'immagine o a un'icona visualizzata sullo schermo. Questo sistema è riuscito a rendere i calcolatori più semplici da utilizzare. Questo nuovo standard, battezzato ***interfaccia grafica utente*** o GUI (Graphical User lnterface), fece ben presto la sua comparsa su tutti i calcolatori. Fu Macintosh a sfoggiare la prima GUI famosa, ma anche Microsoft introdusse ben presto una GUI per i calcolatore IBM e la chiamò **Microsoft Windows**. Dato il successo ottenuto da Microsoft Wíndows, fu predisposta una versione migliorata chiamata Windows 95 seguita poi da Windows 98, ecc...
  Ovviamente, ogni GUI è leggermente diversa dalle altre e i programmi sviluppati per funzionare sotto una GUI non sempre funzionano anche sotto le altre.

Windows: un sistema multitask

Come sappiamo i calcolatori sono strumenti sempre più veloci, ma che sono in grado di eseguire solo un'istruzione alla volta. Questo fatto, in situazioni come ad esempio la videoscrittura, fa si che il microprocessore passi buona parte del tempo in attesa che venga battuto un carattere sulla tastiera. Per ottimizzare il rendimento dei calcolatori è stato studiato un sistema che permette al calcolatore di svolgere contemporaneamente più lavori (**task**). In realtà il calcolatore svolge sempre un azione alla volta ma, prima di passare all'azione successiva del lavoro che sta svolgendo, svolge un azione di eventuali altri lavori che devono essere svolti in parallelo.
Tutti i moderni sistemi operativi, tra cui Windows, permettono lo svolgimento parallelo di lavori.

A fianco un esempio di svolgimento di quattro lavori in parallelo.



Un nuovo approccio alla programmazione

l'utente è al centro

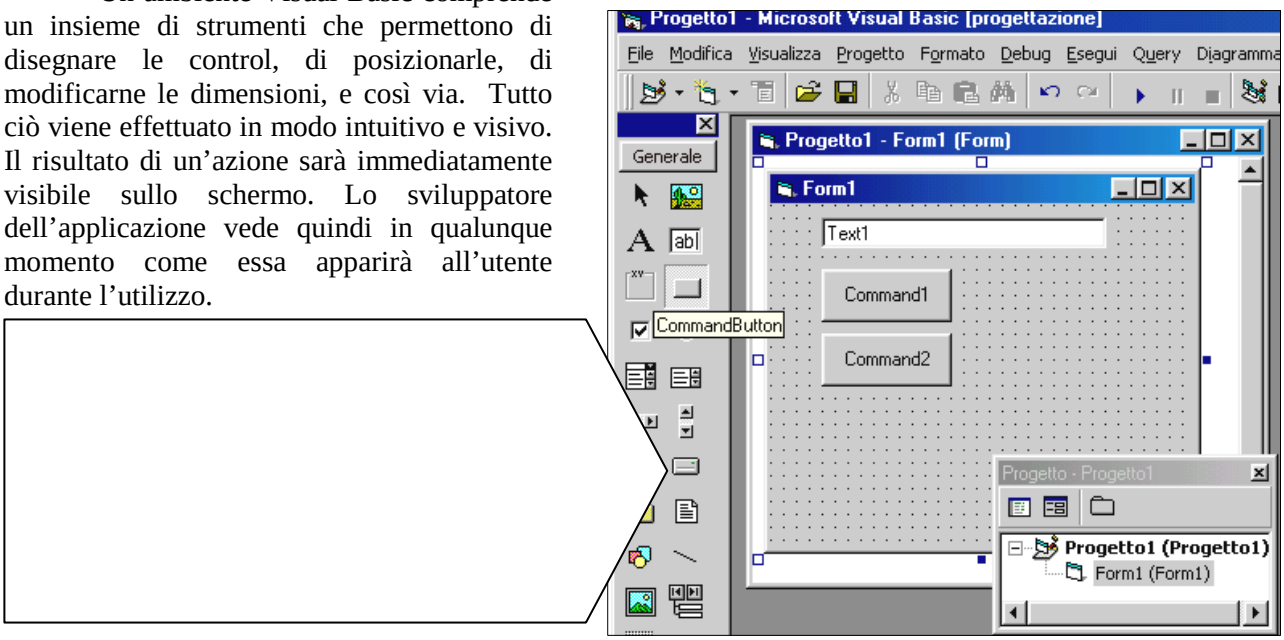
Windows è un'interfaccia utente grafica che pone l'utente al centro. Le applicazioni precedenti (applicazioni DOS) ponevano al centro esse stesse. Lo sviluppatore dell'applicazione si concentrava prima di tutto sull'applicazione, sul contenuto, sulla dinamica e considerava l'utente solo come un fornitore di dati, praticamente al pari di un file. In altre parole si può dire che l'utente è visto solo attraverso periferiche quali schermo e tastiera.

L'approccio di Windows, e quindi di Visual Basic, è diametralmente opposto. Il centro dell'applicazione è rappresentato dall'utente, o più precisamente dall'interfaccia utente. La conseguenza immediata di questo ribaltamento di posizioni è data dal fatto che l'utente può scegliere in qualunque momento cosa fare, per esempio facendo clic sulla parte di schermo che desidera. E' ovvio che tale cambiamento modifica profondamente il modo di concepire e realizzare delle applicazioni.

❑ Il disegno dell'interfaccia

Le applicazioni Visual Basic sono costituite da finestre, come la maggior parte delle applicazioni Windows. La finestra principale dell'applicazione, le finestre secondarie o finestre di dialogo vengono chiamate form. Su tali **form** vengono posizionate le **control** (abbreviazione di finestra di controllo o control window): campi di inserimento testo, pulsanti, liste, eccetera.

Un ambiente Visual Basic comprende un insieme di strumenti che permettono di disegnare le control, di posizionarle, di modificarne le dimensioni, e così via. Tutto ciò viene effettuato in modo intuitivo e visivo. Il risultato di un'azione sarà immediatamente visibile sullo schermo. Lo sviluppatore dell'applicazione vede quindi in qualunque momento come essa apparirà all'utente durante l'utilizzo.



❑ La valorizzazione delle proprietà

Ogni **oggetto** creato, form o control che sia, dispone di proprietà che gli appartengono. Possiamo paragonare gli oggetti di Visual Basic alle specie animali. Le proprietà di un oggetto sono quindi simili alle caratteristiche tipiche di ciascuna specie animale. Alcune sono comuni a più specie (numero di zampe, modo di riproduzione, tipo di pelo, eccetera), mentre altre sono tipiche di determinate specie (lunghezza della proboscide, modo di comunicare, eccetera). Alcune proprietà possono quindi venire applicate a quasi tutti gli oggetti (testo, colore), mentre altre sono tipiche di un determinato tipo di oggetto.

Così come ogni singolo animale di una specie è caratterizzato dai valori attribuiti ad ognuna delle sue caratteristiche, così ogni proprietà di un oggetto ha un valore ad ogni istante. Tali valori vengono inizializzati nel momento in cui l'oggetto viene creato e possono venire modificati in due modi:

- inizialmente in fase di disegno dell'interfaccia
- per mezzo di istruzioni BASIC all'interno del programma.

E' opportuno notare tuttavia che alcune proprietà possono venire modificate e/o lette solo con uno dei due metodi.

Pertanto, dopo aver creato un form o una control, lo sviluppatore può modificarne il valore delle proprietà (testo, colore, eccetera). Alcune proprietà hanno come valore un testo (ad esempio il nome), un numero (larghezza o altezza dell'oggetto) oppure un file (icona).

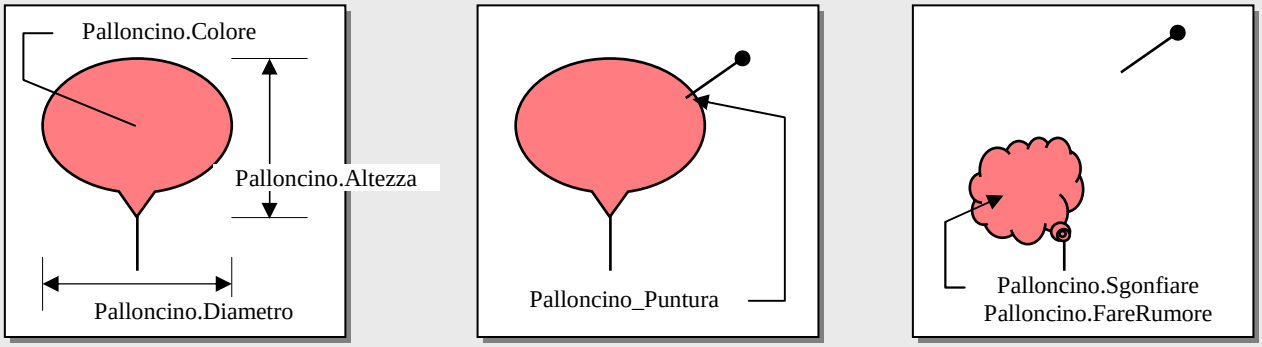
Proprietà, metodi ed eventi

I form e le control di Visual Basic sono oggetti a cui sono associati **proprietà**, i **metodi** ed **eventi**.

*Le proprietà possono essere considerate gli **attributi** di un oggetto, i metodi le **azioni (istruzioni)** eseguite dall'oggetto e gli eventi le corrispondenti **risposte**.*

Per qualsiasi oggetto sono disponibili proprietà, metodi ed eventi specifici. A un palloncino gonfiato con elio, ad esempio, sono associate proprietà per la descrizione di attributi visibili, quali altezza, diametro e colore e proprietà per la descrizione dello stato, ovvero se il palloncino è gonfio o sgonfio, o di attributi non visibili, quali l'età. Per definizione, per tutti i palloncini sono disponibili le stesse proprietà, le cui impostazioni possono tuttavia essere diverse.
A un palloncino sono inoltre associati metodi specifici, ovvero azioni, quali ad esempio il metodo per gonfiare il palloncino (*gonfiare*), il metodo per sgonfiarlo (*sgonfiare*) e il metodo per far sollevare il palloncino (*alzare*). Tali metodi sono disponibili per tutti i palloncini.
Per i palloncini sono inoltre previste determinate risposte a eventi esterni specifici, ad esempio lo sgonfiarsi del palloncino quando viene forato con un oggetto appuntito o il sollevarsi quando viene lasciato andare.

Agli oggetti sono associati proprietà, risposte ad eventi e metodi



Se fosse possibile programmare un palloncino, il codice Visual Basic sarebbe simile a quello riportato di seguito con cui vengono impostate le varie proprietà:

```
Palloncino.Colore = Rosso
Palloncino.Diametro = 10
Palloncino.Gonfiato = Vero
```

Nella sintassi del codice l'oggetto (Palloncino) è seguito dalla proprietà (.Colore), seguita a sua volta dal valore ad essa assegnato (Rosso). È possibile modificare il colore del palloncino nel codice ripetendo l'istruzione e impostando un valore diverso. Le proprietà possono inoltre essere impostate nella finestra Proprietà durante la progettazione dell'applicazione.

I metodi di un palloncino vengono richiamati nel modo seguente:

```
Palloncino.Gonfiare
Palloncino.Sgonfiare
Palloncino.Alzare 5
```

La sintassi è simile a quella delle proprietà, ovvero l'oggetto (un nome) è seguito da un metodo (un verbo). Il terzo esempio include un elemento aggiuntivo, definito *argomento*, che indica di quanto il palloncino sale. Ad alcuni metodi sono associati uno o più argomenti che consentono di descrivere in modo più dettagliato l'azione eseguita.
La risposta del palloncino a un evento può essere simile alla seguente:

```
Sub     Palloncino_Puntura()
        Palloncino.Sgonfiare
        Palloncino.FareRumore = “Bang”
        Palloncino.Gonfiato = Falso
        Palloncino.Diametro = 1
End Sub
```

In questo caso, il codice descrive il funzionamento del palloncino quando si verifica l'evento di foratura (Puntura): viene richiamato il metodo Sgonfiare e quindi il metodo FareRumore con l'argomento "Bang" (il rumore associato all'azione). Dato che il palloncino non è più gonfio, la proprietà Gonfiato viene impostata su Falso e la proprietà Diametro viene impostata su un nuovo valore.
Durante la programmazione di un form o di un controllo di Visual Basic il programmatore dispone del massimo controllo e può impostare le proprietà da modificare, i metodi da richiamare e gli eventi a cui viene fornita una risposta in modo da ottenere l'aspetto e la funzionalità desiderati.

- **La scrittura dei programma**

Anche se le due tappe precedenti permettono di disegnare la presentazione dell'applicazione, si tratta solo della punta dell'iceberg. In realtà l'applicazione può funzionare, ma non è in grado di fare grandi cose. E' quindi necessario scrivere un codice allo scopo di introdurre elaborazioni specifiche.
Il programma è composto da istruzioni in linguaggio BASIC, con estensioni particolari che permettono di tener conto dell'approccio all'oggetto di Visual Basic. Le istruzioni agiscono su delle proprietà o delle variabili. Per esempio è possibile utilizzare un'istruzione per modificare il valore di una proprietà di un oggetto, oppure per verificare lo stato di una variabile.
Le istruzioni del programma devono però essere associate agli oggetti dell'applicazione, form o control.

Tipi di programmazione

La programmazione delle applicazioni di Visual Basic avviene ***per eventi***, in contrapposizione alla programmazione ***lineare*** tradizionale che abbiamo conosciuto lavorando in Liberty Basic.

- **Programmazione lineare**

Se è stato utilizzato un linguaggio di programmazione tradizionale (JUST BASIC, COBOL, C, Pascal, eccetera), allora è stata eseguita una programmazione lineare. Il programma è spesso diviso in tre fasi:

- l'*inizializzazione e l'ingresso dei dati*, durante il quale è possibile preparare lo schermo, dimensionare variabili e matrici, procurarsi i dati dalla tastiera o da file.
- L'*elaborazione*, rappresentata dai calcoli, dai controlli, dai cicli per la gestione di archivi di dati
- l'*uscita dei dati e la terminazione*, che comunica i risultati dell'elaborazione, eseguendo poi le operazioni opposte all'ingresso e cioè il ripristino del sistema.

Paragoniamo lo svolgimento del programma ad un ragno che tesse la sua tela. Qualunque sia la complessità dell'applicazione, se si considera l'esecuzione del programma dal lancio al termine, sarà come se venisse tessuto un solo filo che, una volta sbrogliato, apparirebbe come un unico filo rettilineo.
Questo spiega la denominazione *lineare* attribuita a questo tipo di programmazione.

La struttura generale di una programmazione tradizionale può quindi venire così riassunta:

- Punto di entrata nel programma.
- Punto di uscita.
- Fra i due punti un filo per un'esecuzione dell'applicazione.
- Possibilità di effettuare chiamate a procedure.

- **Programmazione per eventi** (o per oggetti ed eventi)

La programmazione con Visual Basic non corrisponde affatto allo schema descritto precedentemente, pertanto può apparire inizialmente sconcertante. Non esiste più un filo di esecuzione che si svolge dall'inizio alla fine del programma. Un'applicazione Visual Basic è costituita al contrario da un insieme di procedure indipendenti l'una dall'altra.

Una **procedura** comprende delle istruzioni scritte in linguaggio BASIC. Essa viene associata ad un **oggetto**, cioè al form, ad uno degli elementi del form, ma anche ad elementi esterni come la stampante, il robottino Lego, ecc…

La procedura viene chiamata da Visual Basic nel momento in cui ha luogo un **evento** per ***l'oggetto corrispondente***. Se nella procedura incaricata di gestire un tipo di evento per un determinato oggetto non è stato scritto alcun programma, quando l'evento ha luogo non accade nulla di particolare.

Per scrivere il programma di un'applicazione, è opportuno determinare gli eventi al quali si desidera reagire, nonché gli oggetti per i quali l'evento ha luogo. Ciò determina le procedure in cui il programma viene scritto. Si può quindi notare che la programmazione con Visual Basic è esattamente opposta a quella tradizionale. Il conduttore non è più il programma, bensì Visual Basic, e di conseguenza l'utente che si trova all'origine della maggior parte degli eventi.

i modi di funzionamento

Allora, come si costruisce un'applicazione Visual Basic? Visual Basic è un ambiente integrato che permette di disegnare l'interfaccia di utilizzo, di scrivere il programma associato agli eventi, nonché le procedure o le funzioni comuni e di verificare o mettere a punto le applicazioni.

Un ambiente Visual Basic può trovarsi in uno dei tre seguenti modi:

- ***Modo progettazione:*** *è* il modo iniziale durante il quale è possibile creare dei form, inserire delle control, definire le relative proprietà iniziali e scrivere il programma corrispondente agli eventi di tutti questi oggetti grazie ad un editor integrato composto da più finestre.

- ***Modo esecuzione****:* permette di verificare l'applicazione facendola funzionare. Un'applicazione Visual Basic non viene compilata ma interpretata: dopo avere inserito una riga di programma, è possibile eseguire l'applicazione senza dover necessariamente passare attraverso la compilazione o la modifica dei collegamenti. In realtà, Visual Basic genera internamente un *micro-programma* avente lo scopo di velocizzare l'esecuzione: ogni riga viene verificata e interpretata ed il risultato viene disposto in formato binario facilmente leggibile da parte del sistema.

- ***Modo interruzione****:* ha luogo nel caso in cui un'applicazione si interrompa in seguito ad un errore o ad una richiesta esplicita dell'utente. In questa modalità è possibile eseguire la messa a punto dell'applicazione visualizzando il contenuto di variabili o istruzioni, avanzare passo passo e, entro certi limiti, modificare il programma.

Dopo avere creato e verificato un'applicazione in ambiente Visual Basic, è possibile creare un file in codice macchina (*un* ***file eseguibile*** *con estensione .exe*) che sarà autonomo come ogni altra applicazione Windows. Affinché tale file possa funzionare, sarà necessario che sia installato Windows e, in C:\WINDOWS\SYSTEM, sia presente il file MSVBVM60.DLL.
In base all'applicazione realizzata, può essere necessaria la presenza di altri file.

<verifica con 1°test sistemi>

L'AMBIENTE DI SVILUPPO

Un'applicazione Visual Basic è un ambiente integrato che permette la costruzione di applicazioni, nonché la loro verifica e messa a punto. In questa fase mostreremo gli elementi di base relativi all'utilizzo di tale ambiente.

Le finestre di Visual Basic

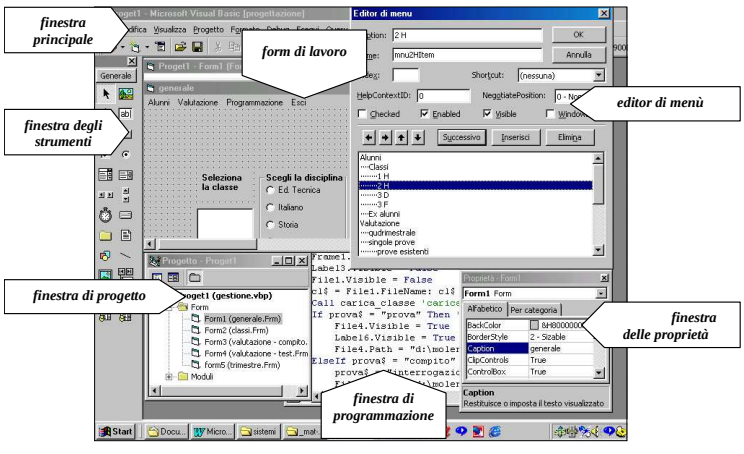
Facendo doppio clic sull'icona che rappresenta l'applicazione Visual Basic il calcolatore carica in RAM il software necessario per il lavoro e apre l'ambiente corrispondente. Attorno alla finestra centrale, la cui barra del titolo contiene il testo "Form1", appaiono numerose finestre. La finestra centrale rappresenta normalmente la finestra principale dell'applicazione che si sta per creare. Le finestre circostanti vengono utilizzate per creare l'applicazione.
Le finestre di Visual Basic sono le seguenti:

- *la **finestra principale***, che comprende tre barre orizzontali: la barra del titolo, come ogni applicazione Windows, la barra del menu che permette l'esecuzione dei comandi e la barra degli strumenti che permette un accesso rapido ai comandi principali. La denominazione *principale è* dovuta al fatto che se tale finestra viene ridotta ad icona tutte le altre finestre vengono nascoste e se la si chiude (selezionando Chiudi dal suo menu di sistema), l'applicazione Visual Basic termina.
- *la **finestra degli strumenti***, che si trova normalmente a sinistra, comprende inizialmente 22 icone nell'edizione standard (vedremo in seguito quando e come possono apparire altre icone). E' possibile accedervi solo in fase di creazione. Quando si fa clic su un'icona, questa viene evidenziata indicando che è stata selezionata.
- *la **finestra di progetto*** comprende la lista dei file che compongono un'applicazione.
- *la **finestra delle proprietà*** comprende una lista delle proprietà dell'oggetto selezionato nel form corrente (form o control) ed i valori corrispondenti.
- *la **palette dei colori*** permette di scegliere un colore da applicare ad una control o ad un form, per esempio il colore dei caratteri di un testo, o il colore dello sfondo. Vi si può accedere solo in fase di creazione.
- *l'**editor di menù*** viene utilizzata per creare il menù da associare ad un form. La si può utilizzare solo in fase di creazione dopo aver selezionato un form.
- *la **finestra di debugging*** consente di verificare o di eseguire delle istruzioni BASIC quando un programma viene interrotto durante il funzionamento. Vi si può quindi accedere solo in fase di esecuzione.
- *i **form di lavoro*** sono le finestre dell'applicazione in corso di progettazione. La finestra centrale "Form1" è un foglio (o form) di lavoro.
- *le **finestre di programmazione*** permettono la visualizzazione o la creazione di istruzioni o dichiarazioni in BASIC.

Queste finestre non sono visibili contemporaneamente. Alcune di esse possono essere rese visibili selezionando la voce corrispondente nel menu ***Visualizza*** della finestra principale, altre invece appaiono solo in fase di esecuzione dell'applicazione. Altre ancora possono essere rese visibili sotto forma di finestre di dialogo, come il visualizzatore oggetti.

Dopo il caricamento, Visual Basic si trova in ***modo progettazione***, è quindi possibile disegnare l'interfaccia dell'applicazione, scrivere il programma, selezionare le proprietà degli elementi, eccetera. Per verificare l'applicazione, selezionare il comando ***Avvia*** del menu ***Esegui***, fare clic sull'icona contenente un piccolo triangolo nella barra degli strumenti, oppure premere il tasto F5 che imposta il ***modo esecuzione***. Un'applicazione creata viene così lanciata, a condizione che non vi siano degli errori. Un'applicazione minima funziona con una finestra principale e basta.
Per interromperla, selezionare ***Chiudi*** nel relativo menu di sistema, fare clic sull'icona contenente una piccola croce nella barra degli strumenti oppure selezionare ***Fine*** nel menu ***Esegui*** di Visual Basic.

Riepilogo di quasi tutte le finestre utilizzabili per progettare un applicazione VB:



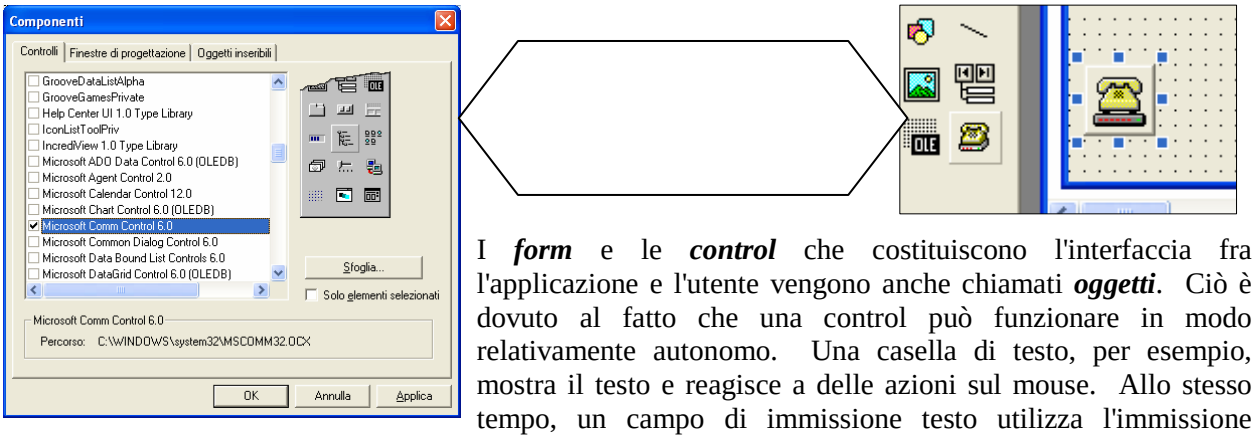
❑ Gli elementi dell'interfaccia (le control)

Un'applicazione Visual Basic è costituita da ***fogli di lavoro*** (***form*** in inglese), che vengono comunemente chiamati finestre. Un form contiene normalmente una barra del titolo, un bordo ed eventualmente una barra del menu e delle barre di scorrimento. Quando si lancia l'esecuzione di un'applicazione, viene generalmente creato un form contenente la barra del menu, chiamato form principale. Durante il funzionamento dell'applicazione possono apparire altri form: si tratta per esempio di form secondari o di finestre di dialogo. Tutto ciò ricorda le altre applicazioni Windows già essere disponibili nel sistema in uso.

I form vengono normalmente utilizzati come riquadri in cui si posizionano altri elementi chiamati **control**. In Visual Basic esistono ventidue tipi di control predefinite (due delle quali sono estensioni), ognuna delle quali corrisponde ad una icona della **finestra degli strumenti** (la prima icona, cioè la freccia, viene utilizzata per selezionare o modificare una control o un form esistente).
Forniamo ora un elenco completo delle control disponibili anche se inizialmente ne useremo solo una piccola parte. Chi di noi vorrà approfondire l'uso del VB per giungere a realizzare applicazioni sempre più professionali imparerà ad utilizzarle tutte. Queste sono le control standard di Visual Basic:

1. ***PictureBox*** *(il controllo immagini)* utilizzato per visualizzare un'immagine, per effettuare stampe di grafica o come riquadro per altre control. E' anche possibile utilizzare *ImageBox* che dispone della proprietà Stretch. Questa proprietà, messa su True, permette un ridimensionamento dell'immagine sulle dimensioni date alla control.
2. ***Label*** *(l'etichetta)* per visualizzare del testo, senza immissione da parte dell'utente.
3. ***TextBox*** *(il campo di immissione testo)* per inserire del testo, su una o più righe.
4. ***Frame*** *(la cornice)* che può raggruppare altre control (soprattutto pulsanti di opzione), o venire utilizzato a scopo decorativo.
5. ***CommandButton*** *(il pulsante di comando)* che permette l'immissione di comandi da parte dell'utente.
6. ***CheckBox*** *(la casella di selezione)* che permette l'immissione di un'opzione di tipo binario (si/no).
7. ***OptionButton*** *(il pulsante di selezione)* utilizzato per selezionare un'opzione fra varie opzioni possibili.
8. ***Timer*** *(l'orologio)* che non è visibile in modo esecuzione, ma permette di generare degli eventi con una periodicità prestabilita.
9. *ComboBox (la lista combinata)* che rappresenta la combinazione di un campo di immissione testo e di una lista semplice.
10. *ListBox (la lista semplice)* per scegliere un'opzione fra varie opzioni, in numero variabile.
11. *HScrollBar (la barra di scorrimento orizzontale).*
12. *VScrollBar (la barra di scorrimento verticale).*
13. *DriveListBox (la lista delle unità)* utilizzata per scegliere una unità a disco.
14. *DirListBox (la lista delle cartelle)* che permette di scegliere una cartella di un'unità disco.
15. *FileListBox (la lista dei file)* per scegliere un file contenuto in una cartella. Queste ultime tre control vengono spesso utilizzate insieme per creare, ad esempio, una finestra di dialogo per la scelta di un file.
16. *Shape (il disegno geometrico)* che permette di disegnare rettangoli e quadrati con o senza angoli arrotondati, ellissi e cerchi.
17. *Line (la linea)* che permette di disegnare un segmento di linea retta.
18. *Image (il disegno) che* è una forma semplificata dell'immagine, permette anche di modificare le dimensioni di un bitmap.
19. *Data (la control dati)* permette di accedere al database.
20. *OLE (il client OLE)* permette l'integrazione di oggetti provenienti da altre applicazioni in un'applicazione Visual Basic.
21. *Grid (la griglia)* permette di definire una tabella visiva di elementi.
22. *CommonDialog (la control finestre di dialogo comuni) per*mette di chiamare le finestre di dialogo standard di Windows.

Come abbiamo visto la finestra degli strumenti può contenere altri elementi corrispondenti ad estensioni che possono venire aggiunte o eliminate dalla finestra degli strumenti. Si tratta di control personalizzate e di oggetti provenienti da altre applicazioni. Una control che noi dovremo aggiungere per poter svolgere le attività di robotica è la *Microsoft Comm Control 6***.** che permette di interagire con una periferica tramite la porta USB. Visto che ci sarà necessario, vediamo già ora come fare.
Sulla finestra degli strumenti clikkare con il tasto destro del file ed aprire il menù dei componenti. Scorrere l'elenco e spuntare sul quadratino di fianco a ***Microsoft Comm Control*** clikkando poi su *Applica* e *Ok*.
Nella finestra degli strumenti comparirà l'icona di un telefono. Dovremo selezionarla per poi portarla sulla form dell'applicazione che prevederà il suo utilizzo.



I ***form*** e le ***control*** che costituiscono l'interfaccia fra l'applicazione e l'utente vengono anche chiamati ***oggetti***. Ciò è dovuto al fatto che una control può funzionare in modo relativamente autonomo. Una casella di testo, per esempio, mostra il testo e reagisce a delle azioni sul mouse. Allo stesso tempo, un campo di immissione testo utilizza l'immissione dell'utente da tastiera per aggiornare il proprio contenuto.

❑ L'ingresso dei dati in Visual Basic

Mentre l'ingresso dei dati da file avviene con le stesse modalità e gli stessi ordini già visti in Just Basic (e L.B.), l'ingresso da parte dell'utente è molto diverso. Non esiste l'istruzione Input che viene sostituita da:

- il ***campo di immissione testo*** *(TextBox)* che permette all'utente, durante l'esecuzione, di scrivere dei caratteri che potranno essere inserti in variabili numeriche o alfanumeriche. Ne vedremo in seguito le modalità di funzionamento
- una apposita finestra denominata ***input box*** che viene organizzata con la seguente sintassi:

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

Parte	**Descrizione**
variabile	Nome della variabile dentro la quale verrà inserito quanto digitato dall'utente
prompt	Espressione di stringa che costituisce il messaggio visualizzato nella finestra di dialogo. La lunghezza massima consentita per l'argomento *prompt* è pari a circa 1024 caratteri e dipende della larghezza dei caratteri utilizzati. Se l'argomento è composto da più di una riga, è possibile separare le varie righe con un carattere di ritorno a capo (**Chr(**13**)**), un carattere di avanzamento riga (**Chr(**10**)**) o una sequenza di ritorno a capo e avanzamento riga (**Chr(**13**)**
titolo	Espressione stringa visualizzata nella barra del titolo della finestra di dialogo. Se si omette l'argomento *titolo*, nella barra del titolo verrà indicato il nome dell'applicazione.
predefinito	Espressione stringa visualizzata nella casella di testo come risposta predefinita se non viene fornito alcun input. Se si omette l'argomento *predefinito*, verrà visualizzata una casella di testo vuota.
posx	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 *posx*, la finestra di dialogo risulterà centrata orizzontalmente.
posy	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 *posy*, la finestra di dialogo risulterà centrata verticalmente a circa un terzo dal bordo superiore dello schermo.

Esempio: prezzovendita = InputBox("Scrivere il prezzo di vendita delle arance in euro", "problema fruttivendolo", 2, 100, 3500)

Vedi esempio in: *fase 2 esempio input output*

❑ L'uscita dei dati in Visual Basic

In V. B. rimane possibile utilizzare l'ordine **Print** per scrivere direttamente sulla form. Bisogna però ricordarsi di aprire la form in modalità grafica utilizzando l'evento ***Form_Paint()*** .
E' anche possibile assegnare, durante l'esecuzione, le proprietà text o caption di una control inserendovi il risultato.
V.B. mette però a disposizione un apposito oggetto per le comunicazioni in uscita che è la ***message box***. La sua sintassi è la seguente:

MsgBox(*prompt***,** *pulsanti***,** *titolo***)**

Parte	Descrizione
prompt	Vedi quanto scritto per Input box
pulsanti	Numero che specifica il tipo di pulsante da visualizzare e lo stile di icona da utilizzare. Se l'argomento viene omesso, il valore predefinito è 0. Per vedere i tipi a disposizione vedere la guida in linea.
titolo	Espressione stringa visualizzata nella barra del titolo della finestra di dialogo. Se l'argomento viene omesso, nella barra del titolo verrà visualizzato il nome dell'applicazione.

Esempio: 'MsgBox Str$(guadagno) + " euro", 48, "il guadagno del contadino è di"

Vedi esempio in: *fase 2 altro es input output*

- **Il primo progetto**

La manipolazione delle control e la creazione della nostra prima interfaccia utente

Per disegnare un oggetto su una form procedete sempre nella maniera seguente:

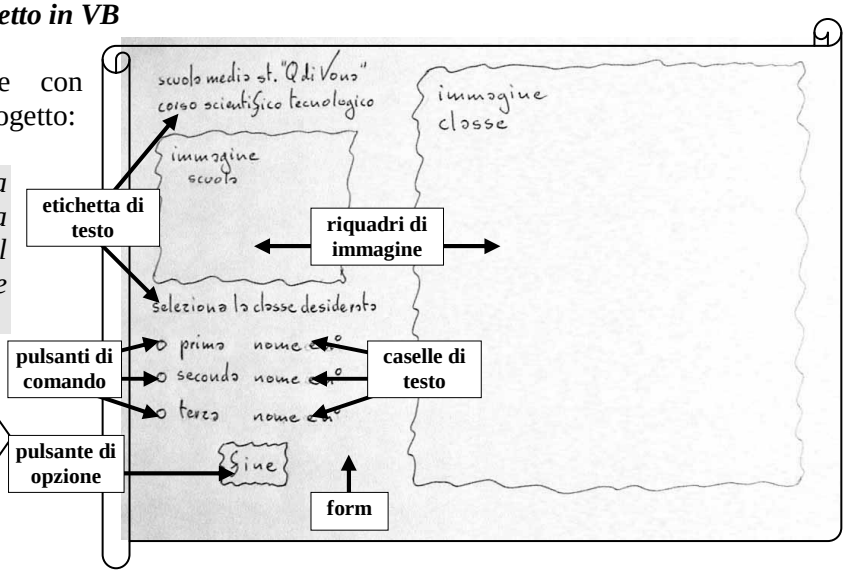
- Fate clic sull'oggetto della barra degli strumenti che desiderate disegnare.
- Posizionate il puntatore del mouse sulla form, nel punto in cui desiderate disegnare l'oggetto.
- Tenete premuto il pulsante sinistro del mouse e trascinate il mouse; questa azione consente di disegnare l'oggetto sulla form.

Realizziamo ora il nostro primo progetto in VB

Per familiarizzare immediatamente con Visual Basic impostiamo un primo progetto:

vogliamo realizzare un programma che mi permetta di selezionare, a richiesta, una delle tre classi del nostro corso mostrandone un'immagine, il nome e il n° di alunni

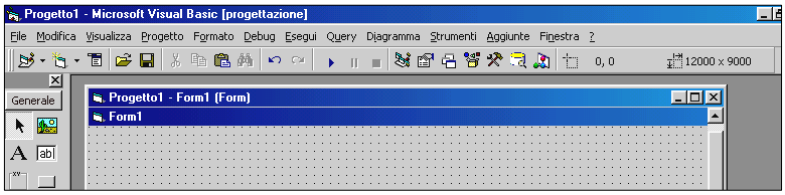
Iniziamo ad impostare il disegno dell'interfaccia su un foglio individuando gli oggetti che dovranno essere utilizzati:

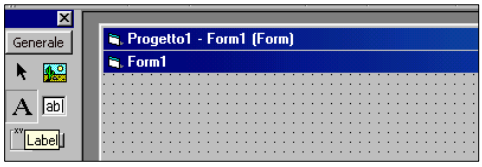


Nel nostro progetto prevederemo un ***etichetta*** *con il nome della scuola* e del corso seguita da *un'****immagine*** *della scuola*. Un'altra ***etichetta*** *inviterà a scegliere la classe desiderata* mentre ***tre pulsanti di opzione*** *consentiranno la selezione*. ***Tre caselle di testo*** *consentiranno di visualizzare nome e numero di alunni della classe scelta* mentre una *fotografia degli alunni comparirà in un grande* ***riquadro di immagine***. Infine ***un pulsante di comando*** *porrà termine all'esecuzione del programma*.
Come si è visto, per visualizzare del testo può essere utilizzata sia l'etichetta di testo (label), sia la casella di testo (text box). Mentre la proprietà Caption dell'etichetta può essere modificata solo in fase di impostazione (e questo vale per tutti gli oggetti che hanno una proprietà Caption), la proprietà Text della casella può essere modificata, come abbiamo già visto, dall'utente durante l'esecuzione e i dati così inseriti possono essere memorizzati dentro delle variabili.

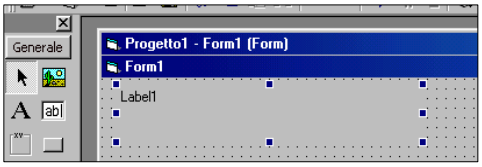
… passiamo ora al disegno della nostra interfaccia

1. In Windows, avviate Microsoft Visual Basic, sempre che non l'abbiate già fatto. Visual Basic visualizzerà un form vuoto intitolato Form1 di cui potremo regolare le dimensioni a nostro piacimento.

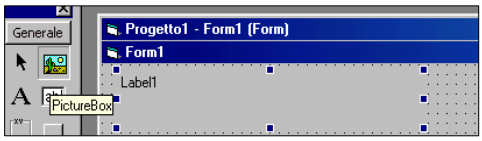




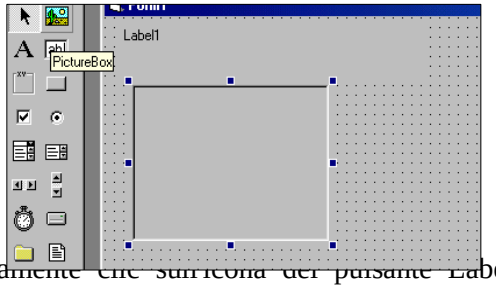
2. Fate clic sull'icona del pulsante Label sulla barra degli strumenti di Visual Basic (vedi a fianco).



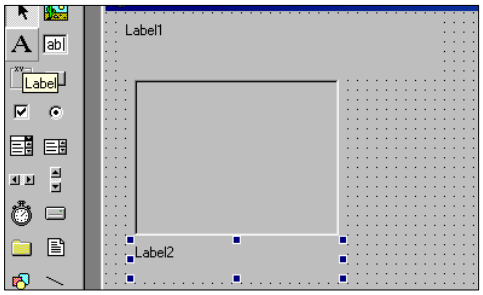
3. Trascinate il mouse in modo che il riquadro dell'etichetta di testo appaia sul form come visualizzato



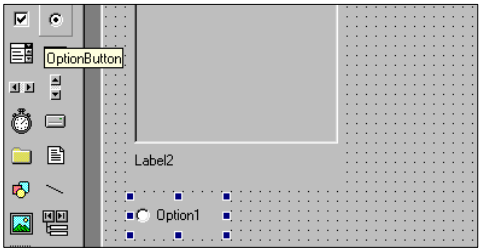
4. Fate clic sull'icona del pulsante Picture sulla barra degli strumenti di Visual Basic (vedi a fianco).



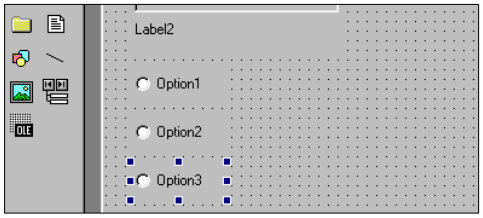
5. Trascinate il mouse in modo che il riquadro di immagine appaia sul form come visualizzato nella figura a fianco.



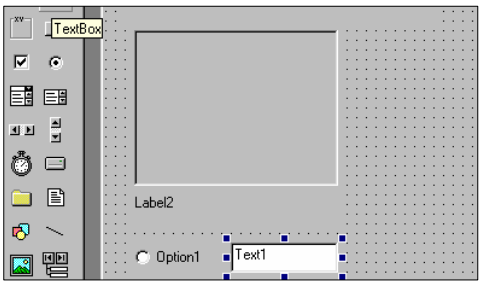
6. Fate nuovamente clic sull'icona del pulsante Label sulla barra degli strumenti di Visual Basic e trascinate il mouse in modo che il riquadro dell'etichetta di testo appaia sul form come visualizzato (vedi a fianco).



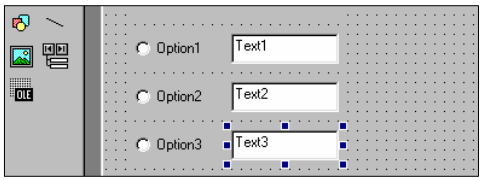
7. Fate clic sull'icona del pulsante Option sulla barra degli strumenti di Visual Basic, quindi trascinate il mouse in modo che il pulsante di opzione assomigli a quanto appare nella figura a fianco.



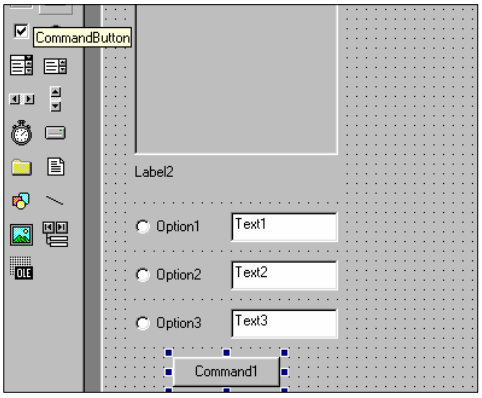
8. Fate ancora clic sull'icona del pulsante Option e disegnate un altro pulsante dedicato alle opzioni. Ripetete nuovamente questo passo. Avete appena creato tre pulsanti di opzione, e il vostro form dovrebbe assomigliare a quello visualizzata nella figura a fianco.



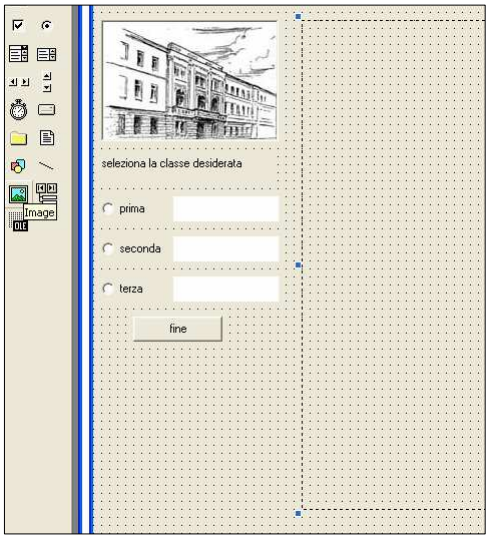
9. Fate clic sull'icona TextBox sulla barra degli strumenti di Visual Basic, quindi disegnate sul form la casella di testo come mostrato nella figura a fianco.



10. Fate ancora clic sull'icona TextBox e disegnate un'altra casella di testo. Ripetete nuovamente questo processo. Avete appena creato altre due caselle di testo, e il vostro form dovrebbe assomigliare a quello visualizzata nella figura a fianco.



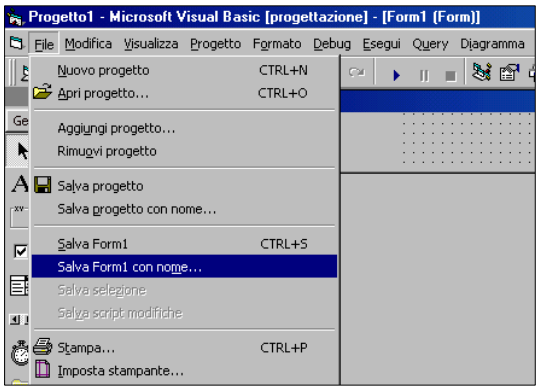
11. Fate clic sull'icona CommandButton sulla barra degli strumenti di Visual Basic e quindi disegnate sul forrn un pulsante di comando come mostrato nella figura a fianco.

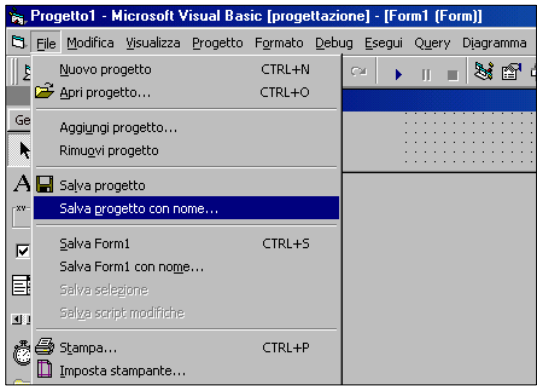


12. Fate clic sull'icona del pulsante Image sulla barra degli strumenti di Visual Basic (vedi a fianco) e trascinate il mouse in modo che il riquadro di immagine appaia sul form. Scegliamo Image perché permette un adeguamento dell'immagine al riquadro, per questo il riquadro dovrà mantenere le proporzioni dell'immagine caricata altrimenti questa verrà deformata.

13. Dal menu File selezionate ***Salva Form1.frm con nome***: verrà così visualizzata una finestra di dialogo Salva file con nome nella quale vi viene chiesto di indicare un nome per il vostro file.

Salveremo il lavoro nella nostra cartella di gruppo





14. Digitate **f_primo_esempio** e selezionate Salva; in questo modo la vostra form viene salvata in un file ***f_primo_esempio.frm.***

15. Dal menu File selezionate ***Salva Progetto con nome***; viene così visualizzata una finestra di dialogo Salva Progetto con nome, nella quale vi viene chiesto come intendete chiamare il vostro progetto.

16. Digitate **primo_esempio** e fate clic su Salva; in questo modo il vostro progetto Visual Basic viene salvato su un file chiamato ***primo_esempio.vbp.***

Avete appena creato un'interfaccia utente Visual Basic generica. Se non vi sembra proprio eccezionale è perché non è ancora stata personalizzata e adattata al vostro programma.
Per personalizzare un'interfaccia utente Visual Basic dovrete definire le ***proprietà*** di ciascun oggetto.

❑ Le proprietà

Ogni oggetto di Visual Basic, sia esso un form o una control, è caratterizzato da ***proprietà.*** Una proprietà rappresenta un attributo, o una caratteristica dell'oggetto. Essa è identificata da un *nome* ed ha un *valore*. Per esempio, numerose control hanno una proprietà il cui nome è Caption e il contenuto corrisponde il testo della control: testo di un pulsante, di una etichetta, della barra del titolo di un form, eccetera. Un'altra proprietà comune alla maggior parte delle control si chiama BackColor. Il suo valore è un numero che identifica il colore di sfondo della control o del form.
L'insieme delle proprietà dei form e di ogni control viene definito da Visual Basic. Non è quindi possibile definire nuove proprietà, ma è possibile modificare il valore di quelle esistenti.
Ciò può venire effettuato in due momenti:

- ***durante la fase di progettazione***
- ***durante la fase di l'esecuzione tramite istruzioni in linguaggio BASIC.***

Qui viene descritta la definizione delle proprietà in fase di impostazione, mentre dell'altra possibilità ce ne occuperemo più avanti.

la modifica delle proprietà

Quando si crea una nuova control in un form come descritto precedentemente, ad ognuna delle sue proprietà viene attribuito un valore iniziale. La finestra delle proprietà, permette di visualizzare le diverse proprietà dell'oggetto, nonché i relativi valori e di modificarli.
Quando si seleziona un oggetto in un form (una control, o se non vi sono oggetti selezionati, il form), la finestra delle proprietà indica le proprietà dell'oggetto. La zona di sinistra contiene i nomi delle proprietà, mentre quella di destra indica i valori corrispondenti. Facendo clic con il mouse su un elemento della lista, il valore corrispondente può venire modificato.
Il valore può venire inserito ammettendo direttamente il nuovo testo se ci si trova nel form, o premendo il tasto F4, oppure facendo clic nel campo di inserimento del valore. Tuttavia, i valori che possono venire immessi dipendono dal tipo di proprietà e vengono indicati dallo stato del pulsante a destra del campo:

- se il pulsante è grigio, inserire direttamente un testo o eventualmente un numero.
- se contiene una freccia, fare clic su di essa per visualizzare una lista di valori da cui selezionarne uno. E anche possibile inserire direttamente la prima lettera del nome. Per esempio, numerose proprietà hanno valore *True o False*, per cui sarà sufficiente inserire "T" o "F".

Se vi sono tre punti di sospensione, facendo clic appare una finestra di dialogo che permette una immissione complementare o una tavolozza colori.
La lista combinata che appare nella parte superiore della finestra delle proprietà permette di selezionare un altro oggetto del form, ad esempio un'altra control.

la scelta dei colore

La scelta del valore di una proprietà corrispondente a un colore è un po' particolare. Può avvenire come descritto precedentemente: scelta della proprietà BackColor per il colore di sfondo o ForeColor per il colore dei caratteri, quindi immissione diretta. Tuttavia il valore da inserire è un numero esadecimale corrispondente alle tre componenti rosso, verde e blu del colore. Facendo clic sul pulsante contenente i tre punti a destra della zona del valore appare una palette del colore. E' anche possibile utilizzare la finestra dei colori che appare selezionando il comando ***Tavolozza colori*** dal menu ***Visualizza***.

Se lo si desidera, tale finestra può rimanere sempre aperta. Per chiuderla, selezionare il comando ***Chiudi*** dal relativo menu di sistema. Essa può venire utilizzata per modificare i colori dello sfondo e dei caratteri dell'oggetto selezionato, anche se la proprietà corrispondente non appare nella zona della finestra delle proprietà. Facendo clic sul due quadrati che appaiono nella parte superiore sinistra della palette è possibile definire la proprietà corrente. La selezione di un colore modifica immediatamente il valore della proprietà.

l'identificazione

Ad ogni oggetto di un'applicazione, form o control, viene attribuito un nome che permette di identificarlo e di designarlo all'interno del programma. Tale nome è il valore di una proprietà chiamata Name. Quando si crea un form o una control, ad essi verrà attribuito un nome per default che può venire modificato come segue:

- selezionare l'oggetto, il form o la control.
- selezionare la proprietà Name nella finestra delle proprietà.
- digitare un nuovo nome.

Il nome deve contenere al massimo 40 caratteri, cominciare con una lettera, essere composto da lettere, da numeri e da caratteri di sottolineatura "_" e non può essere una parola riservata di Visual Basic (tipo Print, For, eccetera. Le lettere possono essere accentate. E' opportuno limitare la lunghezza del nome (meno di 30 caratteri), in quanto il nome di una control o di un form viene associato al nome di un evento per formare un nome di procedura il cui numero totale di caratteri è anch'esso limitato a 40.

Per semplificare la programmazione, i programmatori utilizzano i prefissi indicati nella seguente tabella:

frm per form *(Form): frmFileApri*
pic per disegno *(Picturebox): picdisegno*
txt per campo di inserimento testo *(Textbox): txtNome*
cmd per pulsante di comando *(CommandButton): cmdOK*
sel per pulsante di selezione *(OptionButton): selporta*
lst per lista semplice *(ListBox): lstpaese*
scv per barra di scorrimento verticale *(VScrollBar): scvtassi*
drv per lista di unità a disco *(DriveListBox): drvsorgente*
fil per lista di file *(FileListBox): filsorgente*
lin per linea *(Line): linsep*
grd per griglia *(Grid): grdrisultati*
dat per control dati *(Data): datsocietà*
dbg per griglia con dati *(DBGrid): dbgtabella*
mnu per menu *(Menu): mnuFileChiudi*
etc per etichetta (Label): *etcnome*
cor per cornice *(Frame): corvelocità*
cas per casella di selezione *(CheckBox): casopzione*
cmb per lista combinata *(ComboBox): cmbcittà*
sco per barra di scorrimento orizzontale *(HScrollBar): scocol*
tmr per orologio *(Timer): tmrorologio*
dir per lista di cartelle*(DirListBox): dirsorgente*
geo per disegno geometrico *(Shape): geofondo*
img per immagine *(Image): imglogo*
ole per control OLE *(OLE): oleoggetto*
dbc per lista combinata con dati *(DBCombo): dbcnome*
dbl per lista con dati *(DBList): dbllista*

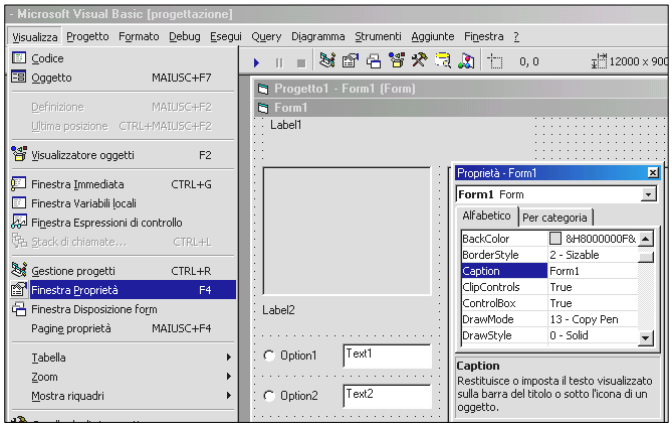
noi, visto il numero più limitato di oggetti utilizzati, useremo solo il carattere iniziale seguito da "barra bassa" e da un nome significativo ad esempio ***p_scuola*** per indicare l'immagine (picture) della scuola.

… passiamo ora a modificare le proprietà degli oggetti della nostra interfaccia

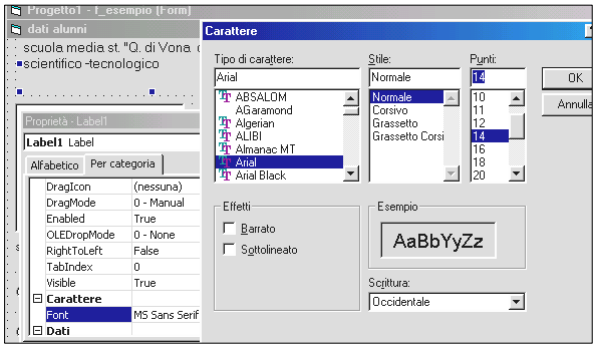
1. Dal menu File selezionate ***primo_esempio.vbp***. Visual Basic caricherà così la form ***f_primo_esempio.frm*** sullo schermo (se la Form è già visualizzata, potrete saltare questo passo).

2. Dal menu *Visualizza*, scegliete *Oggetto* (se il Form è già visualizzato sullo schermo, saltate questo passo).

3. Dopo aver cliccato sulla form apritene la finestra Proprietà selezionando *Finestra Proprietà* dal menu *Visualizza* (vedi fianco).



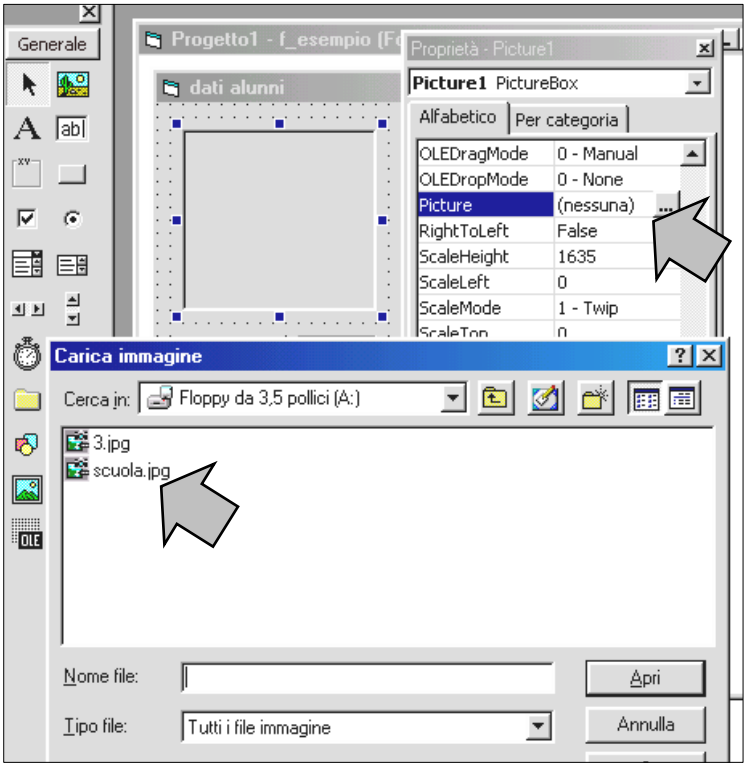
4. Fate clic sulla proprietà ***Caption*** e digitate **dati alunni** che è il nuovo nome che intendiamo dare alla form e che subito comparirà sul bordo superiore blu. Fate poi clic su WindowState e, scegliendo dal menù a cascata, passate da 0-Nornal a 2-Maximized. In questo modo, all'avvio, la form occuperà tutto lo schermo.



5. Fate ora clic su **Label1** per evidenziarla: attorno a essa compariranno dei piccoli rettangoli neri. Fate clic sulla proprietà ***Caption*** e, dopo aver cancellato Label1, digitate l'intestazione scelta. Per evidenziare meglio i caratteri selezionate poi la proprietà Font e compiliamo la finestra di dialogo come illustrato a fianco. Fate poi clic sulla proprietà ***Name*** e digitate **l_intesta**.

6. Fate clic sul riquadro **Picturel**, nella *Finestra Proprietà* e appariranno le sue proprietà.
   - fate poi clic sulla proprietà *Name* e digitate **p_scuola**.
   - successivamente fate clic sulla proprietà *Picture* e dopo sui tre puntini di sospensione. Visual Basic visualizza una finestra di dialogo *Carica Immagine*, come illustrato nella figura a fianco. Selezionate la cartella indicata dall'insegnante e fate doppio click su **scuola.jpg.** L'immagine scelta apparirà subito nel riquadro predisposto.

   Attenzione, è bene preparare con un apposito software l'immagine da caricare nel riquadro regolandone anche le dimensioni. (altezza 5 cm.)



7. Concludete la modifica delle proprietà per il resto degli oggetti in base alla seguente tabella:

Oggetto	***Proprietà***	***Impostazione***
Form (fatto)	Caption	dati alunni
	Name	f_primo_esempio
	WindowState	2-Maximized
Label1 (fatto)	Caption	scuola media st. "Q. di Vona" corso …
	Name	l_intesta
	Font	Arial (norm./14)
Picture1 (fatto)	Name	p_scuola
	Picture	scuola.jpg
Label2	Caption	seleziona la classe desiderata
	Name	l_seleziona
Option1	Caption	prima
	Name	o_prima
Option2	Caption	seconda
	Name	o_seconda
Option3	Caption	terza
	Name	o_terza
Text1	BorderStyle	0-None
	Name	t_prima
	Text	
Text2	BorderStyle	0-None
	Name	t_seconda
	Text	
Text3	BorderStyle	0-None
	Name	t_terza
	Text	
Command1	Caption	fine
	Name	c_fine
	TabIndex	**1**
Image1	Name	i_alunni
	Visible	False
	Stretch	True

Nota bene: nelle caselle di testo è stato cambiato il valore a ***BorderStile*** portandolo lo stile del bordo da ombreggiato (1 – Fixed Single) a nessuno (0 – None) mentre la proprietà ***Picture*** di Image1 non è stata assegnata e la proprietà ***Visible*** è stata messa a False in modo da lasciare, all'avvio, il riquadro di immagine vuoto e invisibile. Lasciamo invece al vostro buon gusto la modifica del colore di sfondo dei singoli oggetti. Per eseguirla si clicca su ***BackColor*** e poi sul menù a cascata che compare di fianco a &H8000000F& (codice esadecimale del colore grigio) e si sceglie sulla Tavolozza.

8. Avviamo, cliccando sul triangolino in alto, il programma e verifichiamo che compaiono form e oggetti tranne l'immagine della classe desiderata con il nome e il numero di alunni. Questi dati compariranno solo cliccando sui rispettivi pulsanti di opzione come vedremo in fase di programmazione.
9. Fermiamo, cliccando il quadratino in alto, il programma e salviamo tutte le variazioni apportate selezionando dal menu File *Save Project* per salvare.

Avete appena definito tutte le proprietà necessarie per la vostra prima interfaccia utente.

- **Il programma**

Anche se dedicheremo la prossima fase ad approfondire il linguaggio Visual Basic anticipiamo ora quanto necessario per completare il nostro progetto.

la scrittura dei programma

Come sappiamo VB si differenzia dal BASIC tradizionale in quanto il programma non viene eseguito in modo sequenziale, ma in risposta a degli ***eventi***. Un evento può avere come origine l'utente (ad esempio la pressione di un tasto della tastiera o un'azione sul mouse), oppure Visual Basic. Inoltre, un evento riguarda sempre un oggetto, un form o una control. Quindi, quando si fa clic su un pulsante di comando, per tale pulsante verrà generato un evento **Click.** Se si fa clic su una lista, verrà generato lo stesso evento **Click,** ma questa volta per la lista. Per ogni evento, Visual Basic chiama una procedura il cui nome è costituito dal nome della control seguito dal simbolo di "barra bassa", quindi dal nome dell'evento. Se, come nell'esempio precedente, il pulsante si chiama *c-fine*, la procedura chiamata da Visual Basic sarà ***c-fine_Click***.
Per le procedure collegate alla form invece utilizza sempre il termine form seguito sempre dal nome dell'evento.
Se invece non è stato scritto alcun codice per la gestione di tali eventi, non accadrà nulla di particolare.
Questo metodo di scrittura del programma, in risposta agli eventi, è all'origine del termine ***programmazione per eventi*** tipico del linguaggio Visual Basic. Questo modo di programmazione in risposta a degli eventi, è direttamente collegato al modo di funzionamento di Windows.

la finestra di programmazione

La scrittura del programma avviene in una finestra di programmazione. Per fare apparire tale finestra:
- fare doppio clic su una control o un form.
- oppure selezionare un form nella finestra di progetto e fare clic sul pulsante CODICE contenuto nella finestra.

Verrà così aperta una finestra di programmazione che proporrà un prototipo della procedura di elaborazione di un evento (inizialmente l'evento gestito più frequentemente) per il form o la control selezionati.

> Una finestra di programmazione è associata ad un form il cui nome di file appare nella barra del titolo. Essa è composta da una barra di selezione (in alto) e da una zona di modifica. Nella parte sinistra della barra di selezione si trova una zona oggetto che permette di selezionare l'oggetto con gli eventi da gestire, mentre nella parte destra si trova una zona di procedura che propone la lista degli eventi disponibili per l'oggetto selezionato nella zona di sinistra. La zona di modifica è un editor di testo classico in cui le istruzioni possono venire inserite. In questa zona delle barre di divisione orizzontale permettono di evidenziale le singole procedure.

Visual Basic effettua un controllo di sintassi ed una formattazione di ogni riga creata ogni volta che si cambia riga. Questa opzione può venire disattivata selezionando la scheda ***Editor*** di ***Opzioni*** dal menu ***Strumenti***. Se viene rilevato un errore, apparirà una finestra di avvertimento. Sarà tuttavia possibile procedere, in quanto l'avvertimento appare una sola volta. Visual Basic analizza ogni riga inserita ed assegna al vari elementi del linguaggio diversi colori, i quali possono venire modificati nella finestra di dialogo ***Opzioni***.
In una finestra di programmazione è possibile inserire:
- ***dichiarazioni di variabili o di costanti utilizzate nelle istruzioni***
- ***istruzioni***
- ***commenti***

Le dichiarazioni e le istruzioni rappresentano gli elementi fondamentali dell'applicazione. Le istruzioni vengono utilizzate da Visual Basic per effettuare delle elaborazioni in risposta a degli eventi. I commenti (REM) permettono di definire i dati utilizzati dalle istruzioni. Se una riga risulta troppo lunga, è possibile suddividerla in più righe, a condizione che le righe che prevedono una continuazione siano seguite dal carattere "_" (sottolineatura).
Nella prossima fase faremo un rapido esame del codice Visual Basic verificando ciò che rimane inalterato e ciò che viene modificato rispetto a quanto noi già conosciamo.

… passiamo ora a scrivere il codice per far funzionare la nostra interfaccia

Prima di iniziare controlliamo l'indirizzo dei file d'immagine delle tre classi (che avremo chiamato *prima*, *seconda* e *terza*) e a cui avremo dato le dimensioni giuste per essere collocati in p_alunni.

1. Dal menu File selezionate **primo_esempio.vbp**. Visual Basic caricherà così la form **f_primo_esempio.frm** sullo schermo (se la Form è già visualizzata, potrete saltare questo passo).

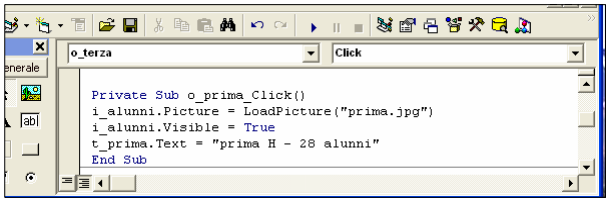
2. Dal menu *Visualizza*, scegliete *Oggetto* (se il Form è già visualizzato sullo schermo, saltate questo passo).

3. Iniziamo ad occuparci della prima. Fate doppio clic sul pulsante di opzione di prima, apparirà la finestra di programmazione che ci propone inizio e fine della procedura o_prima_click:

   Private Sub o_prima_Click()

   End Sub

   completatela con il codice segnato a fianco:



   Da notare l'ordine:

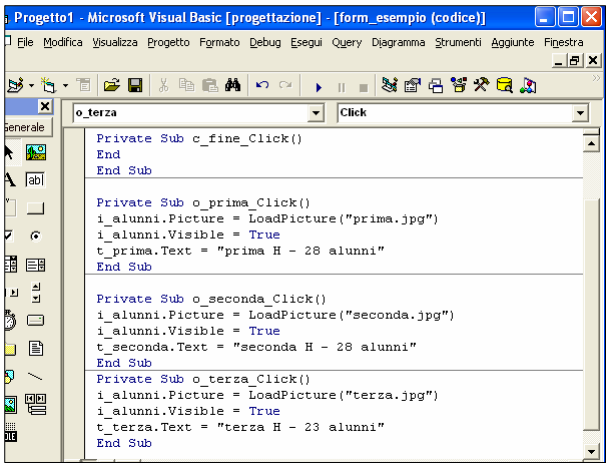
   LoadPicture(*indirizzo e nome del file***)**

   che permette di caricare un file d'immagine dentro il riquadro d'immagine e la sintassi da utilizzare per modificare le proprietà degli oggetti durante l'esecuzione del programma. Come si vede non abbiamo messo il percorso dall'ordine di caricamento dell'immagine. In questo modo il calcolatore dopo aver caricato il programma dalla cartella del gruppo, cercherà i file di supporto nel drive corrente che è proprio la stessa cartella di gruppo.

4. Sull'esempio di quanto fatto con il primo pulsante di opzione, attivare anche le procedure *Private Sub o_seconda_Click()* e *Private Sub o_terza_Click()*

5. Con un doppio clic sul pulsante di comando c_fine, Visual Basic prepara la procedura c_fine_Click dentro la quale scrivere l'ordine **End** che pone fine all'esecuzione

6. Passare ora all'esecuzione di prova che ci permetterà di fare gli ultimi aggiustamenti prima di salvare.



Vedi copia del programma in: *fase 2 primo_esempio*

E' importante ricordare che le immagini inserite durante la fase di impostazione faranno parte del file form. Non sarà dunque necessario conservarne i file di origine. Prima di questo inserimento ci serviremo di un'applicazione grafica per dare alle immagini le dimensioni necessarie. I file inseriti durante la fase di esecuzione dovranno invece essere conservati lasciando immutato l'indirizzo inserito nel programma. Questi file di immagine, come anche eventuali file di dati, sono **file di supporto** e dovranno essere a disposizione dell'applicazione anche quando questa sarà trasformata in un unico file esecutivo
I file di immagine *prima.jpg*, *seconda.jpg* e *terza.jpg* andranno dunque inseriti nella stessa cartella in cui sono collocati i file dell'applicazione.

invio di sequenze di ordini allo stesso oggetto

Quando vi è una sequenza di istruzioni da inviare verso lo stesso oggetto, è possibile utilizzare i seguenti ordini:

- **With** *nome dell'oggetto*

… ordini da inviare all'oggetto

- **End With**

Nell'esempio sottostante una sequenza di ordini inviati verso l'oggetto Printer (stampante) senza e con l'ordine With:

```
Public Sub mio_logo()
    Printer.FontSize = 4 ' mio logo in fondo alla scheda
    Printer.FontItalic = True
    Printer.Print "by Paolo Molena 10/97"
    Printer.FontSize = 10
    Printer.FontItalic = False
End Sub
```

```
Public Sub mio_logo()
    With Printer
        .FontSize = 4 ' mio logo in fondo alla scheda
        .FontItalic = True
        .Print "by Paolo Molena 10/97"
        .FontSize = 10
        .FontItalic = False
    End Whit
End Sub
```

IL LINGUAGGIO VISUAL BASIC

Dopo aver visto come si costruisce un'applicazione Visual Basic, con particolare attenzione al disegno diretto degli oggetti e alla gestione delle loro proprietà, in questa fase approfondiremo le caratteristiche di questo linguaggio con particolare attenzione a quelle istruzioni che fanno effettivamente funzionare l'applicazione. La fase inizia mostrando i file che costituiscono un'applicazione seguita da un'analisi delle modalità in cui VB gestisce le variabili (semplici e complesse) e le costanti. Concluderemo occupandoci delle istruzioni per la gestione dei form e delle control. Lo faremo in modo operativo ampliando l'applicazione realizzata nella fase precedente in modo da inserire un secondo form dedicato alla gestione dei dati degli alunni.

❑ I file di una applicazione

Un'applicazione Visual Basic è sempre costituita da almeno due file: un file form o un file modulo e un file progetto.

File form

L'entità di base di un'applicazione Visual Basic è il form. Ogni form viene memorizzato in un file la cui estensione di default è ***.frm***. E' possibile definire un nome per tale file, ad esempio *dati.frm* o *vendite.frm*.

Il file FRM contiene l'insieme degli elementi associati ad un form. Esso contiene inoltre la descrizione del form e delle relative control, i valori iniziali di tutte le proprietà definite tramite la finestra delle proprietà e il programma inserito nella finestra di programmazione associata al form. Nel file form, il programma viene inserito in un formato tipico di Visual Basic. Un file form ha un formato testo che consente di leggerlo con un editor di testo, di stamparlo o manipolarlo con gli strumenti di gestione dei file. Esso contiene la descrizione del form e delle control con i valori iniziali delle loro proprietà, nonché il programma corrispondente.
Se il form contiene degli elementi binari (per esempio delle icone), questi vengono registrati in un altro file avente estensione ***frx***., il cui nome è uguale a quello del file frm.

Per creare un nuovo form, procedere come indicato di seguito:

- selezionare ***Form*** dal menu ***Inserisci***, oppure fare clic sull'icona corrispondente nella barra degli strumenti.

Viene così creato un form vergine che viene aggiunto alla lista del progetto. Esso verrà salvato su disco solo dopo aver selezionato il comando ***Salva File*** o ***Salva Progetto*** del menu ***File***.
E' anche possibile utilizzare un foglio già esistente allo scopo di integrarlo in un'altra applicazione:

- selezionare ***Aggiungi file*** nel menu ***File***.
- selezionare il form esistente nella finestra di dialogo corrispondente.

File modulo

Oltre al file form, l'applicazione può comprendere un numero qualsiasi di file modulo (con estensione ***.bas***), Questi contengono solo dichiarazioni e procedure generali e non sono associati a nessun oggetto di presentazione, form o control. Le procedure e le funzioni definite in questo tipo di file possono venire chiamate da tutti gli altri file form o modulo, a condizione che non siano dichiarate Private.

Le variabili dichiarate in un modulo possono venire utilizzate nel file modulo se sono state dichiarate con Dim, o in tutti i file modulo e form se sono state dichiarate con Public.

Per creare un nuovo file modulo:

- selezionare ***Modulo*** nel menu ***Inserisci***, oppure fare clic sull'icona corrispondente nella barra degli strumenti.

Alla lista del progetto viene aggiunto un nuovo modulo e viene aperta una finestra di programmazione. Esso viene memorizzato su disco solo dopo aver selezionato il comando Salva file o Salva progetto del menu File.
E' inoltre possibile utilizzare un modulo già esistente al fine di integrarlo in un'altra applicazione. Questo ci sarà necessario nella nostra attività di robotica :

- selezionare Aggiungi file nel menu File.
- selezionare il modulo esistente nella finestra di dialogo.

Estensioni di Visual Basic

Vengono chiamate "biblioteche" archivi di software destinati ad ampliare le possibilità offerte dalle versioni di base dei moderni linguaggi di programmazione. Anche in Visual Basic vi è la possibilità di aggiungere, alle venti control forniti con la versione di base, altro software utilizzando le biblioteche complementari. Queste nuove control si trovano nella finestra degli strumenti di Visual Basic, pertanto in un'applicazione possono venire utilizzate come le control di base. Per aggiungere una control personalizzata:

- selezionare ***Componenti*** dal menu ***Progetto***.
- selezionare un oggetto dalla lista. Essa presenta solo gli oggetti registrati nel registro di Windows. Per aggiungere una control personalizzata ***.vbx***, premere il pulsante Sfoglia nella finestra di dialogo e selezionare il file corrispondente. La control viene così aggiunta nella lista. Dopo aver confermato la finestra di dialogo, la control personalizzata scelta appare nella finestra degli strumenti.

File progetto

Visual Basic utilizza un file progetto (estensione ***.vbp***) per raggruppare i file dell'applicazione. Si tratta di un piccolo file contenente la lista di tutti gli altri file (form e moduli) che costituiscono l'applicazione. Il contenuto del file progetto appare nella finestra di progetto.
Per aggiungere un file al progetto:

- selezionare **Form** dal menu Inserisci per aggiungere un form, oppure fare clic sul pulsante corrispondente nella barra degli strumenti.
- selezionare **Modulo** nel menu Inserisci per aggiungere un modulo, o fare clic sul pulsante corrispondente nella barra degli strumenti.
- selezionare ***Aggiungi file*** nel menu ***File*** per aggiungere un file esistente.

Per eliminare un file:

- selezionare il file da eliminare nella lista del progetto.
- selezionare ***Rimuovi file*** dal menu ***File***.

Il file non viene eliminato dal disco, ma viene semplicemente tolto dalla lista del progetto.

File eseguibile

Per distribuire un'applicazione a degli utenti, è opportuno creare un file eseguibile (estensione ***.exe***). In questo modo non si forniscono i file sorgenti dell'applicazione. Inoltre, l'utente non deve necessariamente installare Visual Basic affinché l'applicazione funzioni (non deve pagare alcun diritto per l'utilizzo di un'applicazione Visual Basic). E tuttavia necessario che diversi file siano presenti sul sistema, in particolare il VB40032.DLL nella directory di sistema di Windows. L'insieme dei file necessari ad un'applicazione dipende dalle estensioni utilizzate (control personalizzate, oggetti, eccetera).
Per creare un file eseguibile:

- Selezionare Crea file EXE nel menu File.
- Inserire il nome del file (per default quello del progetto).

All'applicazione si possono associare anche un'icona e un titolo, facendo clic sul pulsante Opzioni, che verranno utilizzate nel Program Manager.
La finestra di dialogo delle opzioni permette anche l'inserimento di informazioni collegate alla versione dell'applicazione: un numero di versione, una descrizione, informazioni sul copyright e un commento. Queste informazioni appaiono nelle proprietà del file, visualizzate tramite il File Manager.

❑ Le variabili

Come già sappiamo le istruzioni sono composte da verbi che agiscono su delle ***variabili***. Una variabile può essere considerata come una casella in memoria. Essa viene identificata da un ***nome***, che permette di designarla, ed il relativo contenuto si chiama ***valore***.

Il nome della variabile viene scelto da chi scrive il programma, rispettando alcune regole:

- è composto da lettere, da numeri e dal carattere '_' (sottolineatura).
- il primo carattere deve essere obbligatoriamente una lettera.
- si possono usare lettere maiuscole e minuscole ed anche accentate.
- il nome della variabile non può superare i 255 caratteri.
- non si possono utilizzare le parole riservate di Visual Basic (quali **Loop, If, Constant,** eccetera) (in caso contrario usare delle parentesi quadre, ad esempio [Loop]).

Tipi di variabili

Visual Basic permette di utilizzare diversi tipi di variabili che possono venire raggruppate nelle seguenti categorie:

Suff.	**Tipo di dati**	**Dimens.**	**Campo di utilizzo**
%	**Integer** intero	2 byte	da -32 768 a 32 767
&	**Long** intero	4 byte	da -2 147 483 648 a 2 147 483647
!	**Singie** precisione semplice, virgola mobile	4 byte	da -3,402823E38 a -1,401298E-45 per i valori negativi; da 1,401298E-45 a 3,402823E38 per i valori positivi
#	**Double** precisione doppia, virgola mobile	8 byte	da -1,797693134862315D308 a -4,94066D-324 per i valori negativi; da 4,94066D-324 a1,797693134862315D308 per i valori positivi
@	**Currency** intero punto decimale fisso (4 cifre decimali)	8 byte	da -922 337 203 685 477,5808 a 922 337 203 685 477,5807
$	**String**	1 byte per carattere	da 0 a circa 2^31 byte
Nessuno	**Byte**	1 byte	da 0 a 255
,,	**Boolean**	2 byte	True o False
,,	**Date**	8 byte	da 1 gennaio 100 al 31 dicembre9999
,,	**Object**	4 byte	Riferimento ad un oggetto

- *Le variabili numeriche* utilizzate per immagazzinare un numero.
- *Le stringhe di carattere* **(String)** utilizzate per immagazzinare testi, nomi, eccetera. Una sola stringa può contenere oltre 65400 caratteri (nella versione a 16 bit) o 231 byte (circa 2 miliardi di byte nella versione a 32 bit). La lunghezza di una stringa può essere variabile, cioè la dimensione varia in base al contenuto, oppure fissa, in base ad una lunghezza predefiníta.
- Gli oggetti (Object). Visual Basic permette di utilizzare vari tipi di oggetti. Una variabile oggetto immagazzina un riferimento ad un oggetto in 4 byte.

Dichiarazioni

Il linguaggio BASIC presenta un aspetto originale rispetto a numerosi altri linguaggi: esso consente di utilizzare variabili senza imporre una dichiarazione preliminare. Pertanto, il semplice utilizzo di una variabile in una riga di programma genera la sua dichiarazione implicita. E' però consigliabile dichiararle esplicitamente prima dell'utilizzo.

- **dichiarazione esplicita**

La dichiarazione esplicita di una variabile può venire effettuata ovunque all'interno di un programma: nella sezione delle dichiarazioni di un form o di un modulo, in un file incluso, o all'interno di una funzione o di una procedura. Essa ha luogo tramite l'istruzione **Dim, Static, Private** o **Public.** La forma è la stessa in tutti i casi, viene modificata solo la parola chiave.

Ad esempio: *Public lib(26, 600, 28) As String*

Public k, kk, kkk, ks, p As Integer

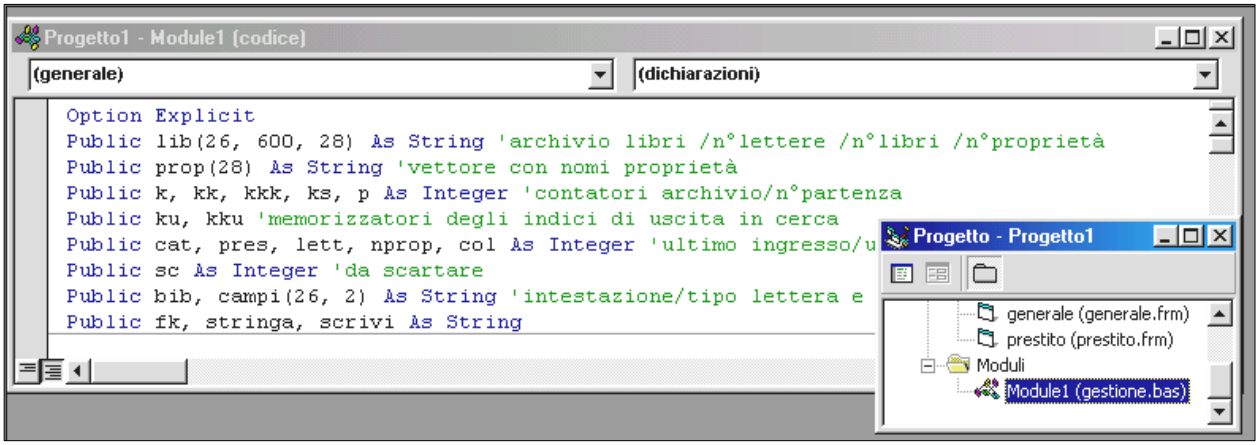
Ogni volta che viene eseguita l'istruzione **Dim,** la variabile corrispondente viene ripristinata a 0 se si tratta di una variabile numerica, alla stringa vuota (una stringa di caratteri senza carattere) se si tratta di una variabile *String***.** Affinché una variabile mantenga il proprio valore in modo permanente, sostituire la parola **Dim** con **Static.** In questo caso, al lancio dell'applicazione la variabile viene inizializzata ai valori precedenti. In seguito, per modificarne il valore, sarà necessario utilizzare un'istruzione di assegnazione.

Per effettuare la dichiarazione esplicita delle variabili è sufficiente inserire nella sezione delle dichiarazioni di ogni form o modulo la riga:

Option Explicit

Quando si dichiara l'opzione *Explicit*, l'utilizzo di una variabile dichiarata non esplicitamente causa un errore. L'uso di questa opzione è vivamente consigliato.

In basso vediamo le dichiarazioni di modulo dell'applicazione realizzata per la gestione della biblioteca scolastica:



- **dichiarazione implicita**

Se una variabile viene utilizzata senza essere stata prima dichiarata (dichiarazione implicita), essa sarà nota solo all'interno della procedura in cui è stata inizialmente utilizzata. Se si fa riferimento allo stesso nome della variabile in un'altra procedura, si tratterà di una variabile differente. E' quindi importante comprendere a fondo l'entità di una variabile, cioè gli elementi del programma in cui essa è nota.

Per fare in modo che una stessa variabile possa venire utilizzata in diverse procedure, sarà necessario effettuare la dichiarazione esplicita. In questo caso importante la parte del programma in cui la variabile viene dichiarata:

- Se la variabile viene dichiarata in una funzione o una procedura, con **Dim**, **Private** o **Static**, essa viene detta ***locale*** rispetto a tale funzione o procedura e non è nota all'esterno. In questo caso, i due termini Dim e Private sono equivalenti.
- Se viene dichiarata nella sezione delle dichiarazioni di un form o di un modulo con **Dim** o **Private**, essa sarà nota in tutte le procedure del form o del modulo, ma non nelle altre.
- Se infine viene dichiarata nella sezione delle dichiarazioni di un modulo con l'istruzione **Public** invece di Dim, essa potrà venire utilizzata in tutti i form e in tutti i moduli. Si tratta quindi di una variabile ***globale***.

❑ Le costanti

Come sappiamo una costante assomiglia ad una variabile in quanto corrisponde ad una casella in memoria. Tuttavia, essa si distingue per il fatto che il suo contenuto viene definito all'inizio e non può venire modificato dal programma. Inoltre, una costante non viene definita da un nome come una variabile, ma semplicemente dal suo valore.

Per facilitare la lettura e la gestione dell'applicazione, si possono utilizzare dei nomi simbolici per definire le costanti. Questi nomi assomigliano, quindi a nomi di variabili, con la differenza che non è possibile modificarne il contenuto.

La dichiarazione di una costante avviene tramite l'istruzione **Const.** Le regole che definiscono l'entità di una costante sono le stesse viste per le variabili, è sufficiente far precedere la parola **Const** da **Public** (nella sezione delle dichiarazioni) affinché la costante sia utilizzabile in altri moduli oltre a quello in cui è stata definita. Per la validità delle costanti così definite vanno rispettate le stesse regole viste con le variabili.

ad esempio con: Const pagabase = 1200 viene definita una costante numerica con valore 1200
ad esempio con: Public Const nomescuola = "Quintino di Vona" viene definita una costante alfanumerica valida per tutta l'applicazione e che dunque potrà essere inserita solo nella sezione Dichiarazioni del modulo.

❑ Le istruzioni nei form e nelle control

Il codice di un programma in Visual Basic è composto da istruzioni che possono essere:
- dichiarazioni per la modifica delle proprietà, di cui ci siamo occupati nella fase precedente
- istruzioni più attive, simili ai verbi di una frase, che noi abbiamo già studiato nell'ambito dello studio delle U.D. Algoritmi e Programmi.

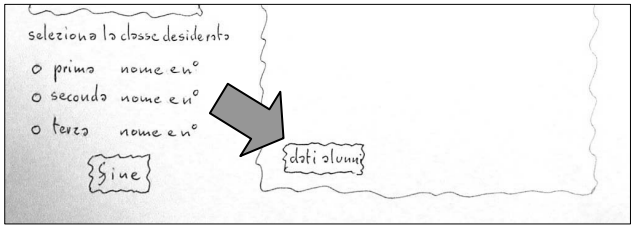
Ripasseremo queste ultime impegnandoci nell'ampliamento dell'applicazione realizzata nella fase precedente. Vediamone il testo iniziale:

Vogliamo realizzare un programma che mi permetta di selezionare, a richiesta, una delle tre classi del nostro corso mostrandone un'immagine, il nome e n° di alunni

ed ora l'ampliamento:

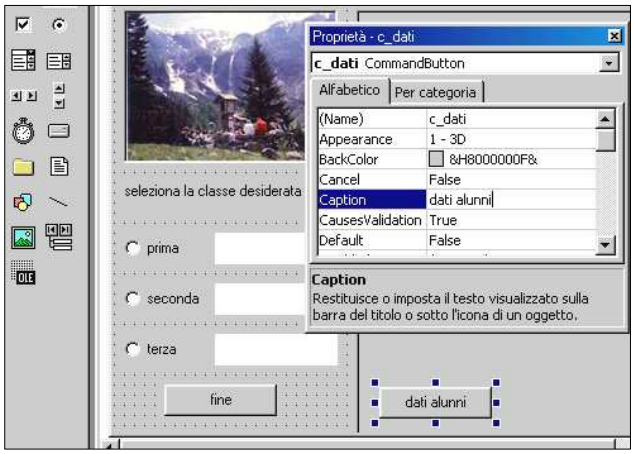
Realizzare un ampliamento programma che mi permetta di visualizzare su un altro foglio (form) i dati degli alunni delle tre classi memorizzati nei file prima, seconda e terza

inseriamo nel nostro progetto su carta un pulsante collocato in basso rispetto all'immagine della classe scelta e che ci possa portare, se selezionato, alla form con l'elenco degli alunni.

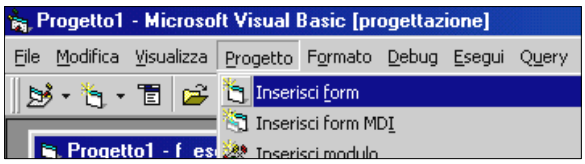


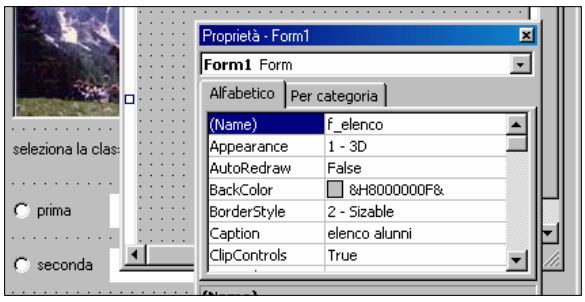
… iniziamo a preparare la nuova interfaccia utente con il nuovo form

10. Dopo aver caricato, con le modalità che già conosciamo, **primo_esempio.vbp** e visualizzato la form **f_primo_esempio.frm** sullo schermo inseriamo un nuovo pulsante di comando nella parte inferiore del riquadro di immagine i_alunni



11. Tenendo evidenziato il nuovo pulsante, ne richiamiamo la finestra delle proprietà. Clicchiamo su *Caption* e, al posto di Command1, scriviamo **dati alunni**. Modifichiamo anche le proprietà *Name* (scrivendo **c_dati**) e *Visible* (mettendola su **False**)





12. Sul menu *Progetto* clicchiamo su *Inserisci form.* Dopo aver confermato la scelta su una ulteriore finestra apparirà sul monitor il nuovo form.

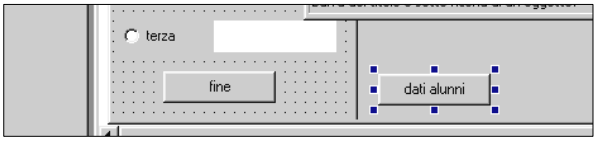
13. Tenendolo evidenziato, ne richiamiamo la finestra delle proprietà e la modifichiamo secondo la seguente tabella (vedi fianco):

Caption	dati alunni
Name	f_elenco
Visible	false
WindowState	2-Maximizated

14. Dobbiamo ora permettere il rientro dalla form f_elenco alla form iniziale. Per questo collocheremo su f_elenco un pulsante in alto a sinistra a cui assegneremo le proprietà a fianco.

Caption	torna al menu iniziale
Name	c_torna

… e completiamo il lavoro inserendo il codice



1. Un doppio clic sul pulsante c_dati ci porta alla finestra di programmazione dove completeremo la procedura proposta nel seguente modo:

Private Sub c_dati_Click()
form_elenco.Visible = True
End Sub

2. E dopo esserci portati sul form f_elenco ed evidenziato il pulsante c_torna scriviamo l'ordine che ci permette di tornare al form iniziale:

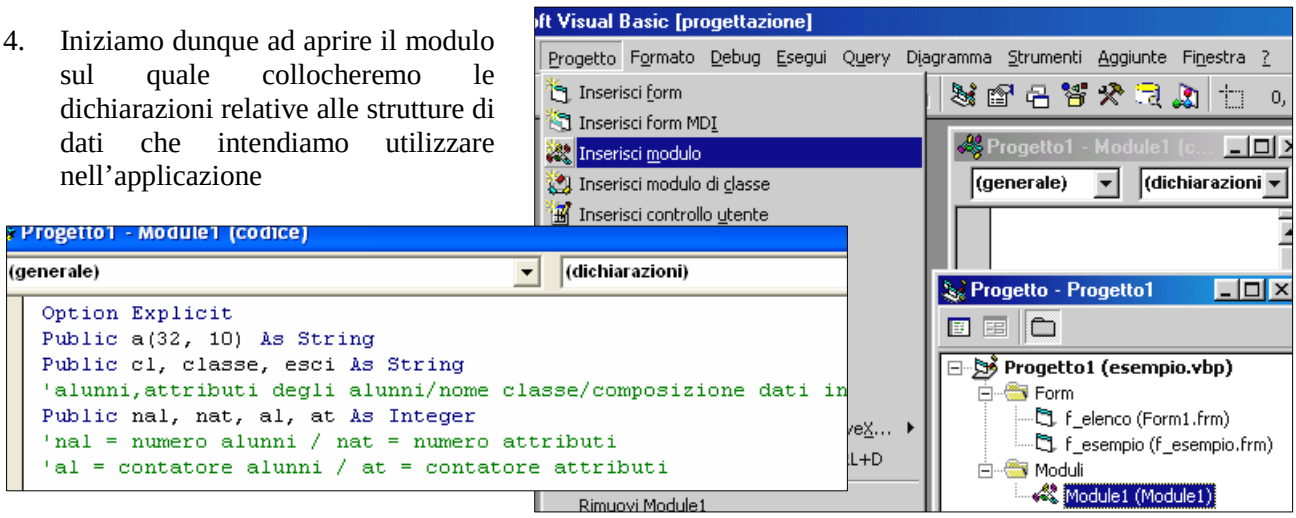
Private Sub c_torna_Click()
form_elenco.Visible = False
End Sub

3. A questo punto è già possibile collaudare l'applicazione, ma sul form f_elenco non comparirà niente visto che non abbiamo dato alcun ordine in proposito. Saranno necessarie un paio di procedure:
   - una per leggere dal file i dati degli alunni della classe scelta e collocarli in una matrice. Questa procedura è di utilità generale, visto che i dati di classi e alunni potrebbero essere utilizzati in tutta l'applicazione. Essa sarà dunque collocata, insieme a tutte le dichiarazioni relative alle strutture di dati da utilizzare, nel modulo
   - un'altra per prelevare dalla matrice e collocare sul form f_elenco quei dati che riterremo utile scrivere. La collocheremo nelle sue dichiarazioni iniziali, mentre richiameremo entrambe le procedure (**Call**) direttamente nella procedura di apertura del form (invece di utilizzare l'evento Form_Load useremo Form_Paint più adatto quando si intende scrivere direttamente sul form con **Print** o con altri metodi grafici.

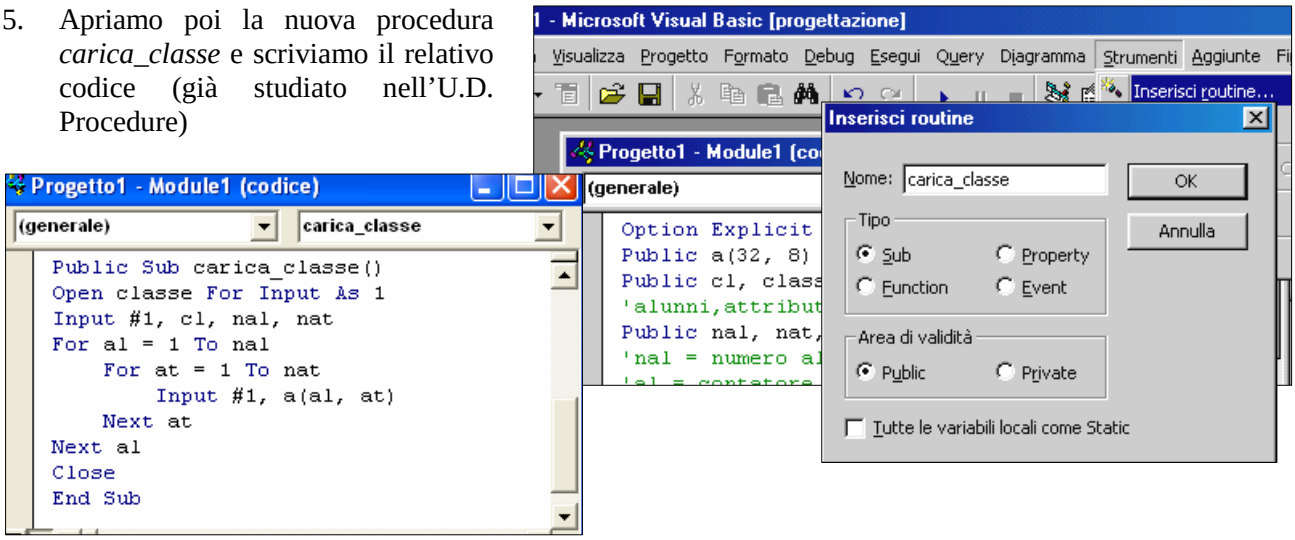
Come le immagini delle tre classi anche i file di testo contenenti i dati degli alunni sono **file di supporto** e dovranno essere a disposizione dell'applicazione anche quando questa sarà trasformata in un unico file esecutivo

I file di testo *prima.txt*, *seconda.txt* e *terza.txt* andranno inseriti nella stessa cartella in cui sono collocati i file dell'applicazione. Per farli gestire correttamente dal programma è necessario conoscere con precisione come sono stati memorizzati i dati nei file.

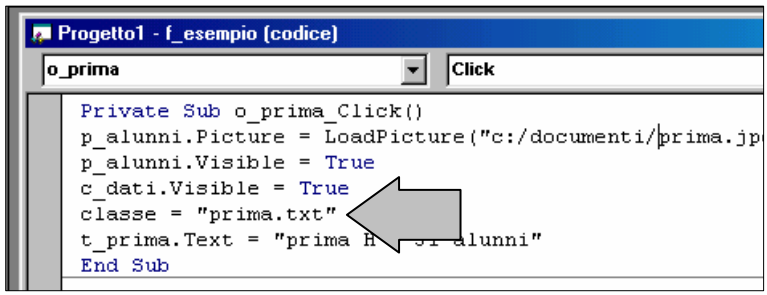
4. Iniziamo dunque ad aprire il modulo sul quale collocheremo le dichiarazioni relative alle strutture di dati che intendiamo utilizzare nell'applicazione



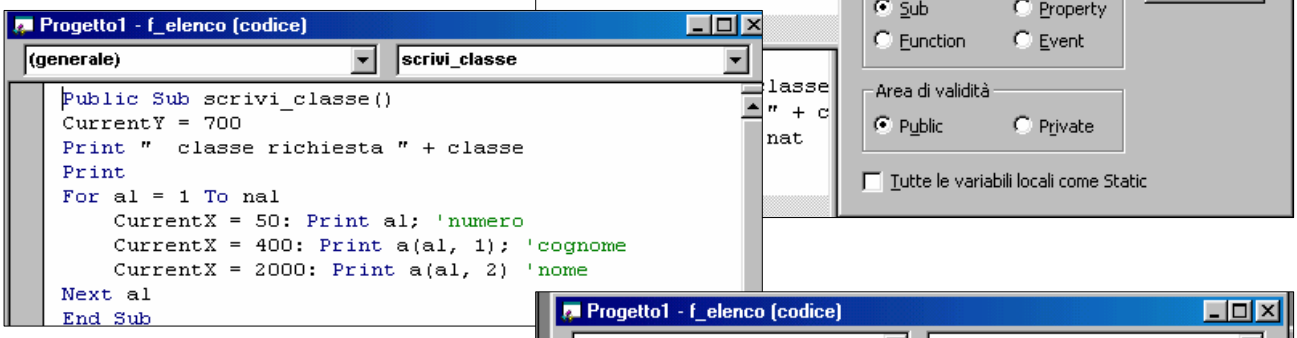
5. Apriamo poi la nuova procedura *carica_classe* e scriviamo il relativo codice (già studiato nell'U.D. Procedure)



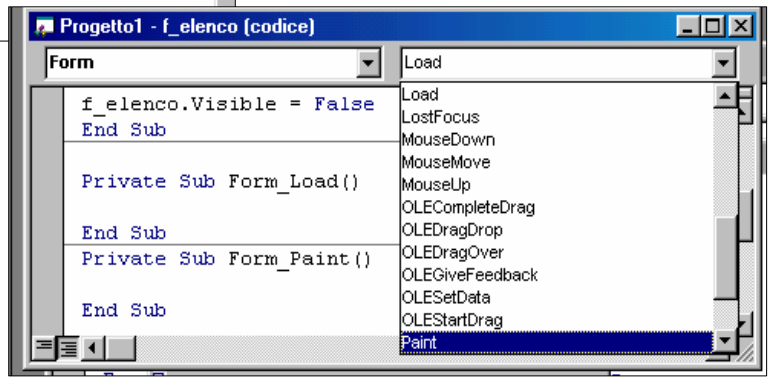
6. Per aprire con un unica procedura il file scelto è necessario che dentro la variabile classe ci sia il nome del file da caricare. Andranno così modificate le tre procedure relative alla scelta della classe (a fianco quella della prima con l'assegnazione della variabile classe). Completare il lavoro su *o_seconda* e *o_terza*.



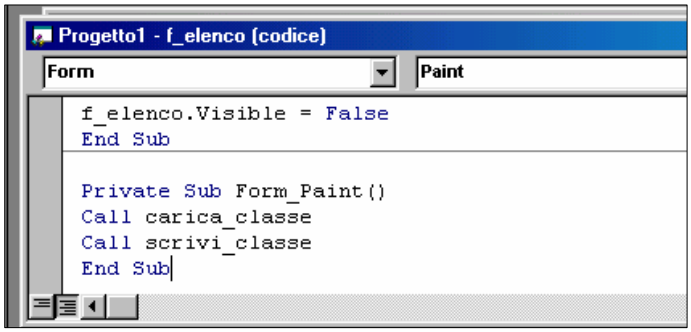
7. Dopo aver reso attiva la finestra di programmazione di f_elenco, apriamo nuovamente la finestra *inserisci routine* per aprire la procedura scrivi_classe e successivamente scriviamo la procedura.



8. Concludiamo il lavoro evidenziando il form f_elenco e facendovi un doppio clic. VB vi proporrà la procedura Form_Load. Noi invece la cancelleremo dopo aver aperto, grazie al menù a cascata degli eventi, una procedura Form_Paint. In questa verranno inseriti i richiami.



9. Passare ora all'esecuzione di prova che ci permetterà di fare gli ultimi aggiustamenti prima di salvare definitivamente il nostro lavoro.



… e ora "mettiamo in vendita" la nostra applicazione

Come sappiamo chi realizza software a livello professionale lo fa per guadagnare. Noi non possiamo pensare di vendere la nostra applicazione ma potremmo anche volerla portare a casa e volerla vedere sul nostro calcolatore. In questo caso, prima di trasferire tutto il software utile su un supporto removibile, procederemo alle seguenti operazioni:

- creare una nuova cartella e raggrupparvi i file dell'applicazione e i file di supporto
  *creeremo sul supporto la cartella **nuovo_esempio** e vi trasferiremo i file necessari*
- se tutto funziona creare un file ***.exe*** (che chiamerete *esempio*) rimuovendo poi dalla cartella tutti i file dell'applicazione (sulla cartella resterà il file eseguibile più i file di supporto)
  per creare il file .exe seguire le istruzioni date nelle schede precedenti
- mettere nella cartella una copia del file MSVBVM60.DLL che potrà essere copiato dalla cartella WINDOWS/SYSTEM di un calcolatore su cui sia stato installato Visual Basic
- portare la nuova cartella su un supporto removibile (chiavetta, cd, ecc) per poi, eventualmente, trasferire il tutto sull'altro calcolatore. L'applicazione partirà al lancio del file *.exe*

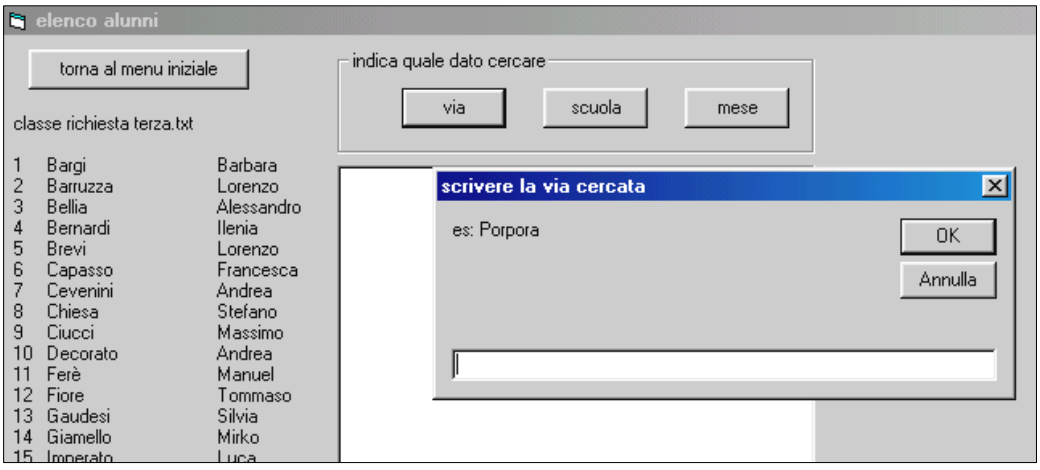
❑ Programmi di ricerca

… in una matrice bidimensionale

Sinora vi siete limitati a ricopiare ciò che ci hanno suggerito le schede. Ora invece il gruppo è chiamato ad arricchire l'applicazione mettendo alla prova le conoscenze acquisite.
Adattate dunque il programma sinora realizzato e preparate sul form elenco una cornice che contenga alcuni bottoni che permettano di svolgere le seguenti ricerche:

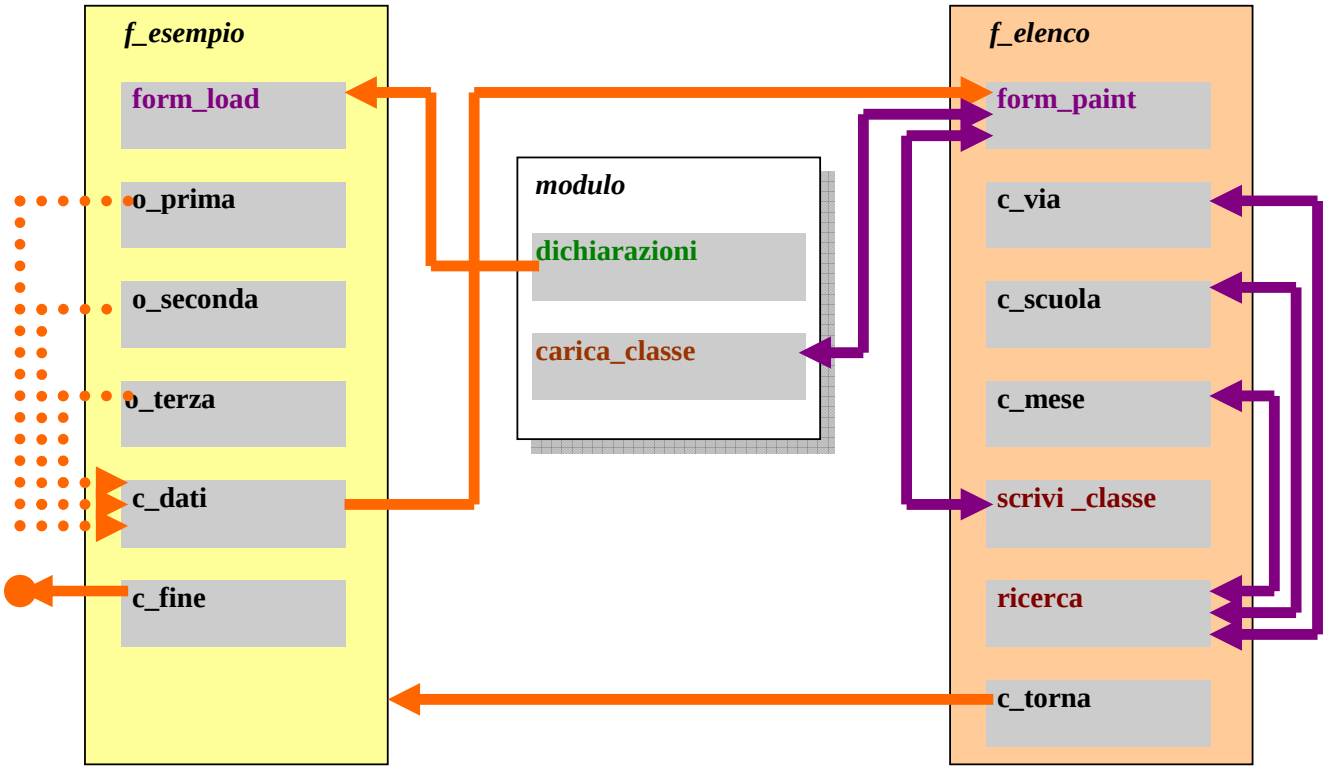
- dato il nome di una via scrivere cognome e nome degli alunni che abitano nella stessa via
- dato il nome di una scuola elementare di provenienza scrivere cognome e nome degli alunni che provengono da quella scuola
- dato il numero del mese di nascita (es: 10 = ottobre) scrivere cognome e nome degli alunni che sono nati in quel mese

per l'input utilizzerete un unico input box (che chiamerete **cerca**) e per l'uscita dei dati della ricerca organizzerete una Text.box che chiamerete t**_dati**.

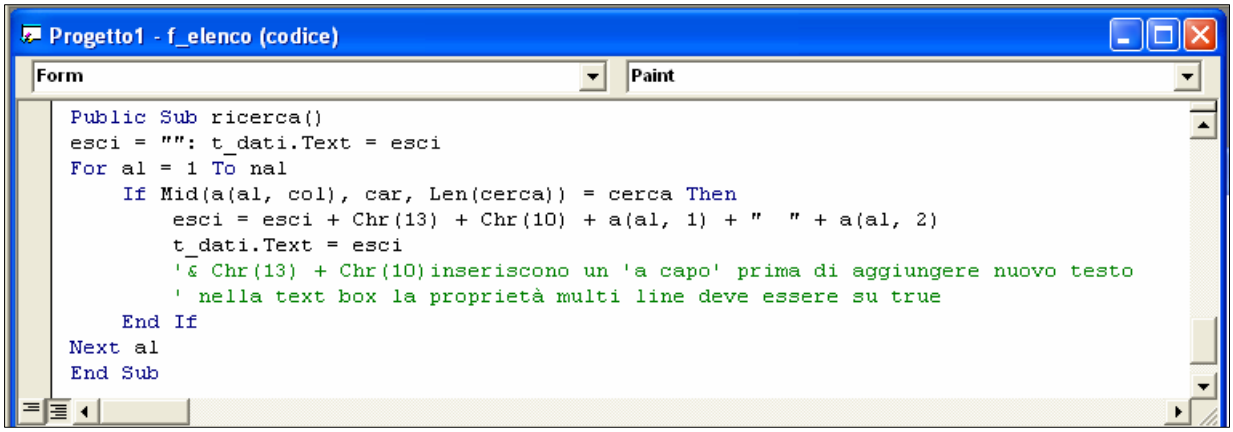


Quelle richieste, sono ricerche che abbiamo già svolto in JB. Abbiamo anche imparato che queste tre ricerche possono essere svolte utilizzando un'unica procedura (che chiameremo **ricerca**) in cui siano resi parametrici i dati che cambiano. I tre bottoni (che chiameremo **c_via**, **c_scuola**, **c_mese**) conterranno le chiamate alla procedura **ricerca** seguite dai valori da assegnare ai dati parametrici.

Il seguente schema rappresenta le relazioni tra gli oggetti presenti nel programma:



Durante la ricerca i dati trovati non possono essere assegnati singolarmente alla proprietà **Text** di **t_dati** man mano che vengono trovati. Infatti l'ultima assegnazione cancellerebbe quella precedente. E' così necessario comporre i dati precedenti più il nuovo in una variabile e assegnarne il contenuto alla proprietà Text.
Per inserire un "a capo" dopo ogni cognome e nome si utilizzano i loro codici ***Chr(13)+Chr(10)*** seguendo le regole di sintassi indicate nell'esempio seguente:



… in una matrice tridimensionale

Sappiamo come è fatta una matrice tridimensionale ma non abbiamo potuto lavorarci in JB. Ora in Visual Basic possiamo finalmente utilizzare questa importante struttura di dati memorizzando i dati di tutte e tre le classi del corso H ognuna in un suo foglio (prima=foglio 1 / seconda=foglio 2 / terza=foglio 3).

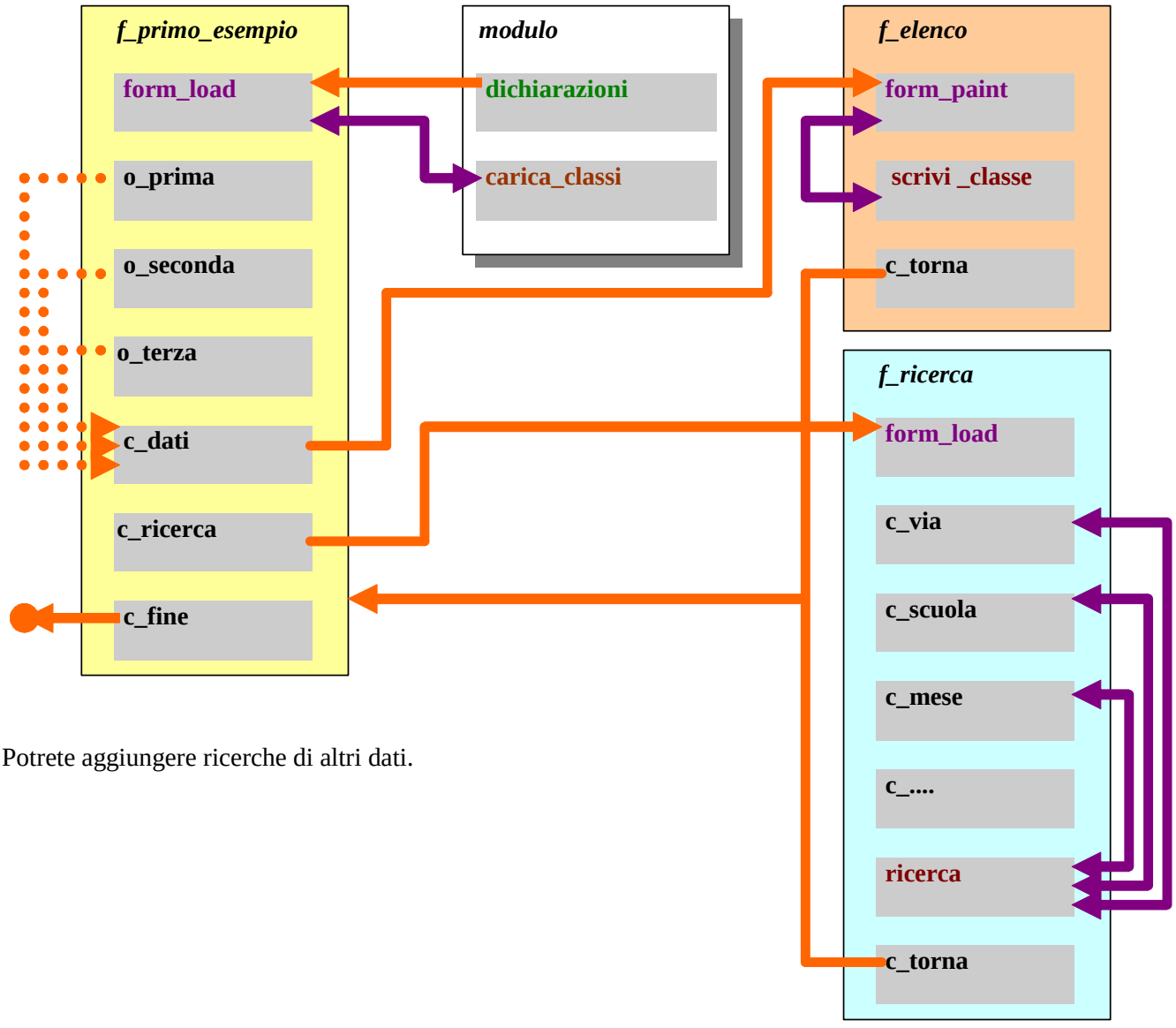
Possiamo dunque partire dal programma già fatto ma cambiandone completamente la logica di funzionamento.

- nella dichiarazione andrà modificato il dimensionamento della matrice che diventa tridimensionale e poi bisogna aggiungere una matrice bidimensionale per i campi di testa
  Public a(32,10) As String
  diventa:
  Public a(3, 32, 10) As String
  Public dat(3, 3) As Integer

- è bene che i dati vengano letti subito, all'avvio della form, in modo che siano tutti presenti nella RAM durante l'esecuzione del programma. Ovviamente il caricamento dei dati sarà modificato perché dovrà caricare nella matrice tridimensionale i dati delle tre classi. Sarà chiamato **carica_classi**.

- sulla form **f_elenco** è bene lasciare la visualizzazione dei dati degli alunni (non solo nome e cognome) mentre trasferiremo su una nuova form (che potrete chiamare **f_ricerca**) la ricerca dei dati, modificata in modo da farla sulla matrice tridimensionale.

Sulla form **f_esempio** dovremo collocare un nuovo pulsante che conduca a **f_ricerca**.
Ecco lo schema del nuovo programma:



Potrete aggiungere ricerche di altri dati.

ROBOTICA IN VISUAL BASIC

La robotica (cenni storici)

I primi studi per creare strumenti che imitino le capacità di manipolazione dell'uomo risalgono agli antichi greci. Nel settecento l'orologiaio Pierre Jacquet-Droz stupiva i suoi contemporanei costruendo dei piccoli "automi antropomorfi", cioè automi costruiti con le sembianze dell'uomo. Il più celebre è lo "scrivano", un automa raffigurante un ragazzo seduto ad una piccola scrivania e capace di scrivere fino a quaranta caratteri di un testo. Si trattava di automi funzionanti con gli stessi principi degli orologi meccanici.

Il termine **robot** (dallo slavo robota che significa lavoro) viene introdotto nella lingua inglese nel 1921 dal commediografo céco Karel Capek. In un suo dramma satirico (I robot universali di Rossum) i robot sono macchine che assomigliano agli uomini, ma che lavorano instancabilmente. Nel dramma i robot erano stati costruiti a scopo di profitto per sostituire gli operai, ma verso la fine essi si rivoltano contro i loro creatori, distruggendo l'intera razza umana. L'opera di Capek è grandemente responsabile di alcuni *luoghi comuni* ancora oggi esistenti, tra cui il pensare ai robot come a macchine umanoidi dotate di intelligenza e di personalità individuali.

I primi studi che hanno portato ai moderni robot industriali (cioè controllati da microprocessori elettronici) avvengono nel periodo immediatamente successivo alla seconda guerra mondiale e portano alla realizzazione, alla fine degli anni quaranta, di manipolatori meccanici in grado di operare su materiali radioattivi ripetendo i movimenti del braccio di un uomo collocato ad opportuna distanza. Nella metà degli anni cinquanta, il controllo di questi manipolatori divenne elettronico (erano oramai in circolazione i primi calcolatori). Mentre questi robot erano costruiti in modo da ripetere sempre la stessa sequenza di operazioni, nel 1959 viene realizzato il primo robot industriale, cioè un robot che essendo programmabile può adattarsi alle mutevoli esigenze della produzione industriale. Infine negli anni sessanta vengono realizzati i primi sensori; il robot acquista dunque la capacità di adattarsi all'ambiente circostante.

E' negli anni settanta che i robot industriali iniziano ad essere utilizzati in larga scala in USA e in Giappone. Contemporaneamente nasce la **robotica**, la scienza che si occupa dello studio dei robot e che è in rapida espansione.

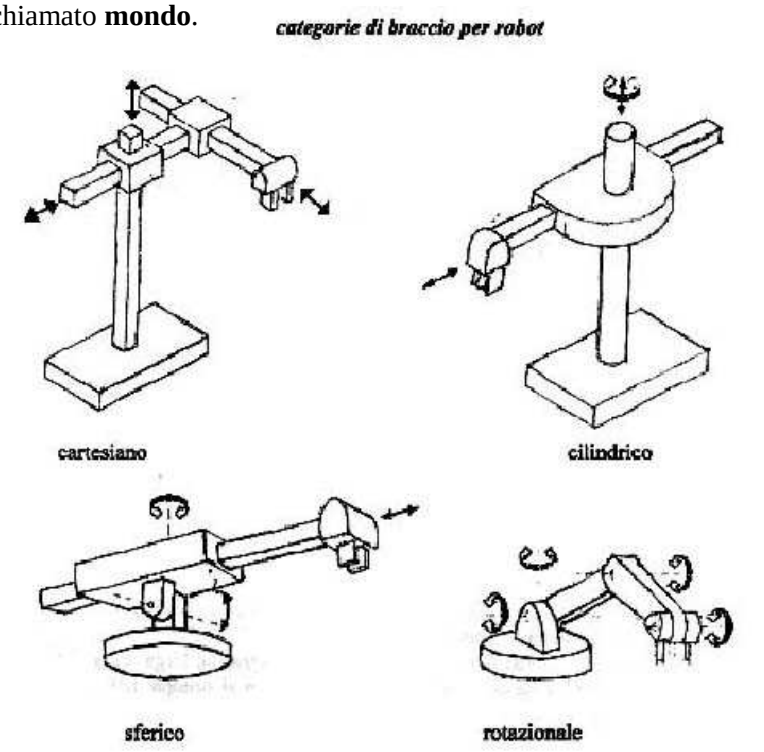
I robot oggi

Un robot industriale è dunque un manipolatore riprogrammabile progettato per spostare / assemblare materiali per mezzo di movimenti variabili programmati per l'esecuzione di un dato numero di compiti. Esso è progettato per raggiungere un pezzo della lavorazione situato nel suo spazio di lavoro. Tale spazio viene definito come sfera di influenza del robot e chiamato **mondo**.

Molti robot industriali sono largamente usati nei compiti di fabbricazione e di assemblaggio, nella manipolazione di materiali, nella saldatura a punti o ad arco, nell'assemblaggio di parti, nella verniciatura a spruzzo, nel carico e scarico di macchine ecc.

Questi robot appartengono ad una delle quattro categorie fondamentali in relazione alla definizione del moto:

- ***coordinate cartesiane***
  (tre assi lineari X, Y, Z)
- ***coordinate cilindriche***
  (due assi lineari X, Z e uno rotazionale)
- ***coordinate sferiche***
  (un asse lineare X e due rotazionali)
- ***coordinate rotazionali***
  articolate o antropomorfe
  (tre assi rotazionali)



Conclusioni

Nel lungo itinerario che ha portato l'uomo a realizzare strumenti sempre più in grado di sostituirlo nelle attività della vita quotidiana il punto di arrivo è uno strumento che sia in grado di sostituirlo interamente. Questo significa:

- *saper memorizzare nuove procedure* > ***controllo***
- *manipolare nuovi oggetti e materiali utilizzando anche i necessari strumenti e operare e spostarsi in vari tipi di ambienti* > ***meccanica***
- *procurarsi le informazioni necessarie per lo svolgimento del lavoro* > ***percezione***

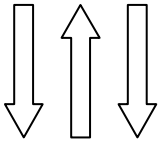
Per ora si è ancora lontani dal creare strumenti in grado di lavorare a questo livello di prestazioni. L'ostacolo principale è proprio nell'estrema complessità dell'ambiente in cui vive l'uomo, nella grande quantità di situazioni, a volte imprevedibili, in cui ci si può trovare, nella necessità di prendere decisioni inconsuete per far fronte ad eventi inconsueti. Per questi motivi l'uso del robot è possibile solo in ambienti delimitati, a contatto con situazioni prevedibili.

Se perciò è ancora lontano il tempo il momento in cui vedremo robot camminare per le strade e dialogare con noi, ***già oggi possiamo vedere intere produzioni industriali, dall'ingresso delle materie prime al collaudo dei prodotti finiti, svolte in impianti robotizzati, con una presenza dell'uomo limitata al controllo e alla manutenzione. Grandi quantità di prodotti possono così essere assicurate con la presenza di pochissimi uomini***.

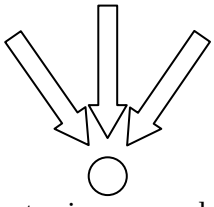
Se le fosche previsioni di Capek non si stanno avverando, è però vero che l'affermazione della robotica nelle moderne industrie sta cambiando completamente il modo di produrre. Sta oramai sparendo l'operaio di linea, cioè l'operaio addetto alle produzioni ripetitive. Essi in passato costituivano non solo la parte più consistente della forza lavoro nelle fabbriche ma anche la parte più attiva, sia in senso sindacale che politico. Le rivendicazioni operaie hanno profondamente inciso nelle vicende storiche dalla metà dell'ottocento sino ai giorni nostri essendo state alla base della nascita di partiti politici e di ideologie, di rivoluzioni, di grandi cambiamenti sociali.
Oggi la scomparsa delle "classe operaia" e la sua sostituzione con un'elitè di tecnici specializzati, sta causando profondi cambiamenti nel nostro modo di produrre, di vivere e anche di pensare.

Il "mondo", i mondi esclusivi e non (con situazioni di parallelismo, concorrenza e conflitto)

Per programmare il robot è fondamentale la definizione del **mondo**, cioè della spazio in cui esso è destinato ad operare. Esso può essere ***esclusivo*** quando questo spazio è unicamente di sua competenza. ***Mondi non esclusivi*** possono essere determinati dalla presenza dell'uomo o di altri robot. Visto che il comportamento dell'uomo non è sempre prevedibile con certezza, la sua presenza nello spazio di azione del robot determina una situazione di grave difficoltà difficilmente superabile.
Abbastanza comune è invece la presenza di più robot nello stesso mondo. In questo caso, pur avendo ognuno di essi un compito ben preciso, chi realizza il software dovrà farsi carico della loro gestione complessiva. Potranno determinarsi queste situazioni:



una situazione di **parallelismo** quando ogni robot può svolgere il proprio compito senza interferenze da parte degli altri.



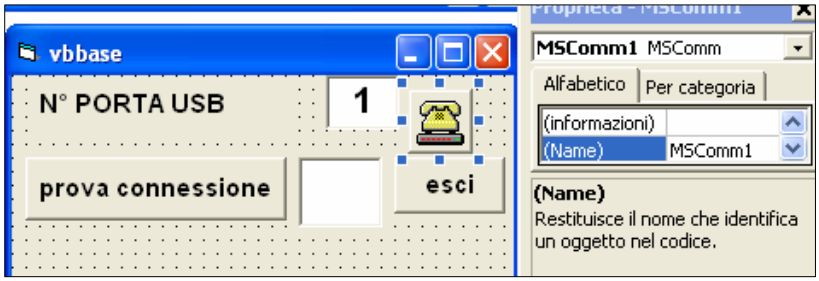
una situazione di **concorrenza** quando i robot hanno un compito comune per realizzare il quale ognuno dovrà svolgere una propria parte.

Le una situazione di concorrenza non è ben programmata si possono determinare situazioni di **conflitto** tra robot (alcuni robot impediscono ad altri di portare a termine il loro compito) e di stallo (i robot si fermano a causa di una situazione non prevista).
Per ora noi ci occuperemo di situazioni più semplici mentre l'anno prossimo, lavorando in Visual Basic, potremo realizzare ambienti in cui più robot saranno in grado di scambiarsi messaggi e di lavorare intorno a progetti comuni.

COME LAVORA UN ROBOT (motori e sensori)

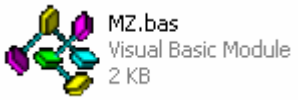
La connessione

MZ è una periferica "intelligente", è cioè in realtà un piccolo computer dotato di un proprio microprocessore e in grado di lavorare in modo autonomo. Dispone dunque di un proprio ***firmware*** che può essere reinstallato dal sito di NUZOO. Può comunque lavorare sotto il controllo del nostro computer, controllo che avviene grazie al collegamento con una porta USB. In Visual Basic, per dialogare con una periferica collegata alle porte USB è necessario inserire un apposito controllo: il ***Microsoft Comm Control***.

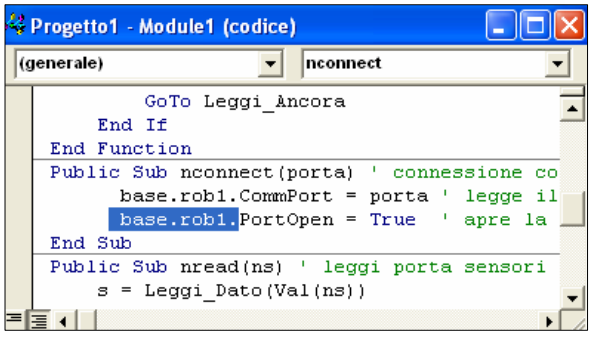


Questo controllo si aggiunge agli altri oggetti presenti sulla form. Di default viene chiamato ***MSComm1*** ma in tutti i nostri esempi il nome è stato sostituito con ***rob1***.
Una volta inserito questo controllo possiamo comunicare con MZ ma utilizzando gli ordini standard che Visual Basic mette a nostra disposizione per comunicare con delle periferiche.

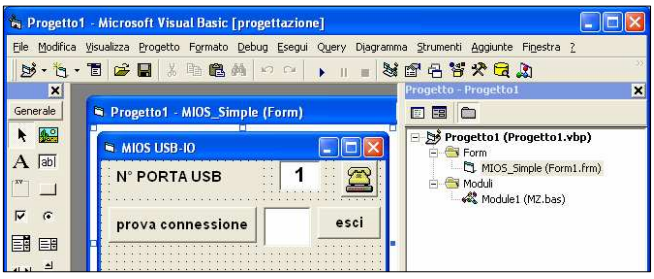
Come abbiamo fatto in Liberty Basic, anche in Visual utilizzeremo apposite procedure che semplificano il nostro lavoro di robotica.
Queste sono raggruppate nel file *MZ.bas* che va inserito come modulo all'interno del nostro programma con la procedura già studiata. Questo file svolge gli stessi compiti che svolgeva il sw in coda ai programmi di liberty.



E' importante tenere presente che nel modulo *MZ.bas*, ogni ordine rivolto alla nostra scheda di robotica deve essere preceduto dal nome dell'oggetto e dal nome della form su cui l'oggetto è collocato.
In tutti i nostri esempi abbiamo chiamato ***base*** la form e' come abbiamo già visto, ***rob1*** l'oggetto.
Se vogliamo utilizzare il modulo MZ.bas senza fare modifiche, anche nei nostri progetti dovremo usare gli stessi nomi.

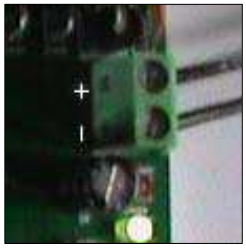


Noi potremo utilizzare i nomi delle procedure per controllare i motori e per ricevere i valori dei sensori. Nel caso dei sensori vengono utilizzate due variabili, che dunque sono state rese globali, per ospitare il valore letto dal sensore (s) e per rendere binario tale valore (v) a seconda se il valore letto supera una determinata soglia (che è modificabile).
A destra possiamo osservare la form e la finestra di progetto del programma ***vbbase*** che contiene il modulo MZ.bas. Questo modulo, da questo progetto, può essere inserito in nuovi progetti.



E' bene ora attivare il collegamento con MZ.
E' bene farlo seguendo questa procedura:

- accendere il calcolatore
- inserire nella porta USB il cavo di MZ.
  Sulla piastra centrale di MZ si accende una luce verde.
  Segnala che sta ricevendo il segnale dal calcolatore tramite il cavo USB
- alimentare MZ utilizzando un trasformatore con uscita di corrente continua a 12volt e avvitando i due fili negli appositi ingessi (ricordare che la posizione dei fili non è indifferente). L'accensione del led sottostante indica che la scheda viene alimentata correttamente.



Se vi sono segnali che indicano problemi nel collegamento togliere e rimettere il cavo USB.

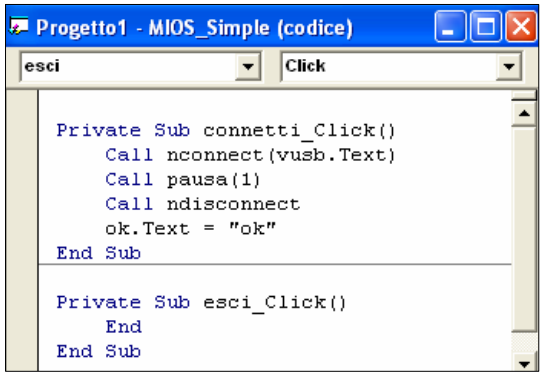
In ogni programma di robotica saranno presenti i seguenti ordini (già visti in L.B. con qualche piccola differenza di sintassi):

```
call nconnect ( _ ) 'tra parentesi va inserito il n° della porta USB ut ilizzata
call nwrite (9,1) 'serve per rendere possibile l'utilizzo dei motori
'---

'---
call ndisconnect ' chiude la connessione on MZ
end
```

Mentre la gestione dei motori e dei sensori avverrà con gli ordini:

```
call nwrite ( _,_ ) 'tra parentesi il numero della porta del motore (0-7) e il valore trasmesso (1 per accendere / 2 per spegnere)
call nread ( _ ) 'tra parentesi il numero della porta del sensore (0-7)
```

Tra gli esempi troviamo il programma ***vbbase*** di cui abbiamo già visto la form ed ora, a destra, vediamo il codice. Esso è realizzato per poter sperimentare l'efficienza della connessione.
Il click su *prova connessione* attiva la connessione e, dopo un breve pausa, la disconnessione.
E' importante notare che il n° della porta USB su cui è collegato MZ viene inserito dall'utente nella casella di testo *vusb* e letto da V.B. con Call nconnect (vusb.Text).



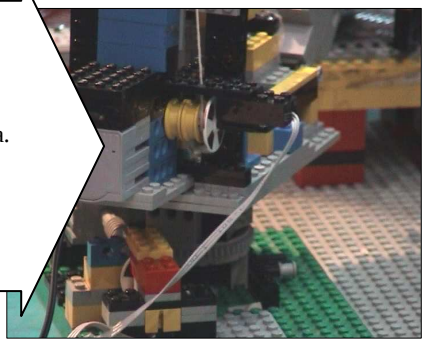
Per effettuare una prova sulle letture del sensore possiamo avviare il programma ***prova-sensore***. La form viene avviata con l'evento *Paint* visto che dobbiamo utilizzare l'ordine *Print* per scrivere sulla form. Dopo aver inserito in due textbox il numero della porta USB e poi quello della porta del sensore, si clicca su ***prova sensore*** e il calcolatore effettua 50 letture scrivendo sulla form i valori analogico e digitale di ogni lettura (contenuti nelle variabili s e v).

Per effettuare un test più completo sul buon funzionamento di MZ avviare il programma ***test***.
Sulla form vi sono tre textbox nelle quali vanno inserite i numeri della porta USB, della porta motori e della porta sensori. Poi clikkare su ***connetti*** (il computer legge la porta e si connette). Quando si clicca su *leggi*, nella label *value_Text* compare il valore letto dal sensore. Infatti nella procedura ***leggi*** viene prima effettuata la lettura dal sensore dichiarato con Call nread (vsens.Text).

Subito dopo con VALUE_Text.Caption = Str(s) viene inserito nella label, dopo averlo reso alfanumerico, il valore analogico letto (che è nella variabile s). Ad ogni click verrà effettuata una nuova lettura.
Con la procedura ***muovi*** vengono prima abilitate le porte dei motori Call nwrite(9, 1) e poi viene data corrente alla porta stabilita con Call nwrite(vmot.Text, 1). La procedura ***ferma*** toglie la corrente alla porta. Per terminare l'attività si può cliccare su esci.

Prima di chiudere il programma (End), viene chiusa la connessione. Prima della chiusura, per sicurezza, vengono disabilitate le porte dei motori con Call nwrite(9, 0); questo serve nel caso l'utente abbia dimenticato di fermare uno dei motori.

La "ruota contagiri" sul braccio robotico presentato a Robotica 2009.
Il sensore ottico collocato davanti alla ruota bianco-nera permette di rilevare il passaggio dalla fascia bianca a quella nera.
Il programma permette di accendere il motore e di spegnerlo quando è stato raggiunto il numero di variazioni necessario per raggiungere la posizione desiderata.

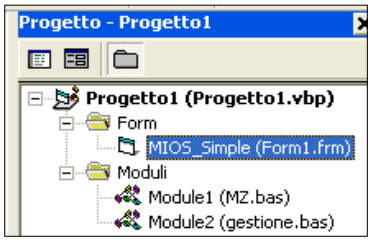


Il modulo **gestione.bas**

Nella nostra attività in Liberty Basic avevamo già realizzato due procedure per facilitare la realizzazione di programmi di robotica più complessi. La procedura ***muovi*** serve per gestire l'attività di un motore controllato grazie ad un sensore su una rotella contagiri. La procedura ***attendi*** serve per gestire l'attesa di un sensore. Attesa che può essere di una o più variazioni (con 0 variazioni si esce subito).
In Visual Basic è possibile collocare queste procedure in un apposito modulo (gestione.bas), che si aggiunge così a MZ.bas, lasciando invece il programma principale nel codice della form.
Il programma ***muovi-fisso*** è utile per controllare il lavoro di un motore collegato ad una rotellina contagiri.
Sulla form sono indicati i valori su cui è stato impostato il programma: porta USB n°13, porta motori n°0, porta sensori n°0, n° variazioni chieste n°4.
Se abbiamo esigenze diverse dovremo modificare i valori all'interno del codice.

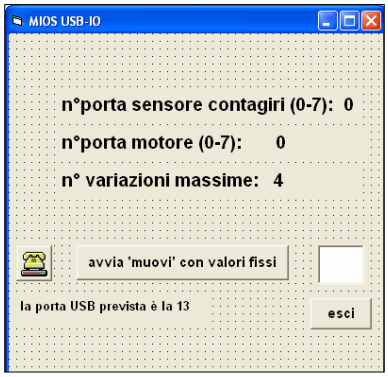
```
(generale)                                  muovi
'---  muovi
Public Sub muovi(pm, ns, vmax)
Call nread(ns)
a = v: b = v: Var = 0
Call nwrite(pm, 1)
Do While Var < vmax
        Do While a = b
                Call nread(ns): b = v
    Var = Var + 1: a = b
    MIOS_Simple.CurrentY = 2500
    MIOS_Simple.Print Var; "  ";
Loop
Call nwrite(pm, 0)
End Sub
' ---attendi
Public Sub attendi(ns, vmax)
Call nread(ns)
a = v: b = v: Var = 0
Do While Var < vmax
        Do While a = b
                Call nread(ns): b = v
        Loop
    Var = Var + 1: a = b
    Print Var
Loop
End Sub
```

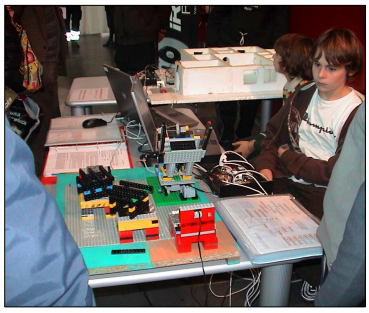
in alto a destra il modulo ***gestione.bas*** che, insieme a ***MZ.bas*** costituisce la base dei nostri progetti di robotica.
A destra\la finestra di progetto di ***muovi-fisso*** mentre sotto abbiamo il codice e la form.
I valori 13, 0, 0, 4 possono essere cambiati nella finestra del codice.



Il programma ***muovi-param*** prevede invece che i valori vengano forniti dall'utente tramite le textbox e inseriti con le modalità già viste nel programma *test*.

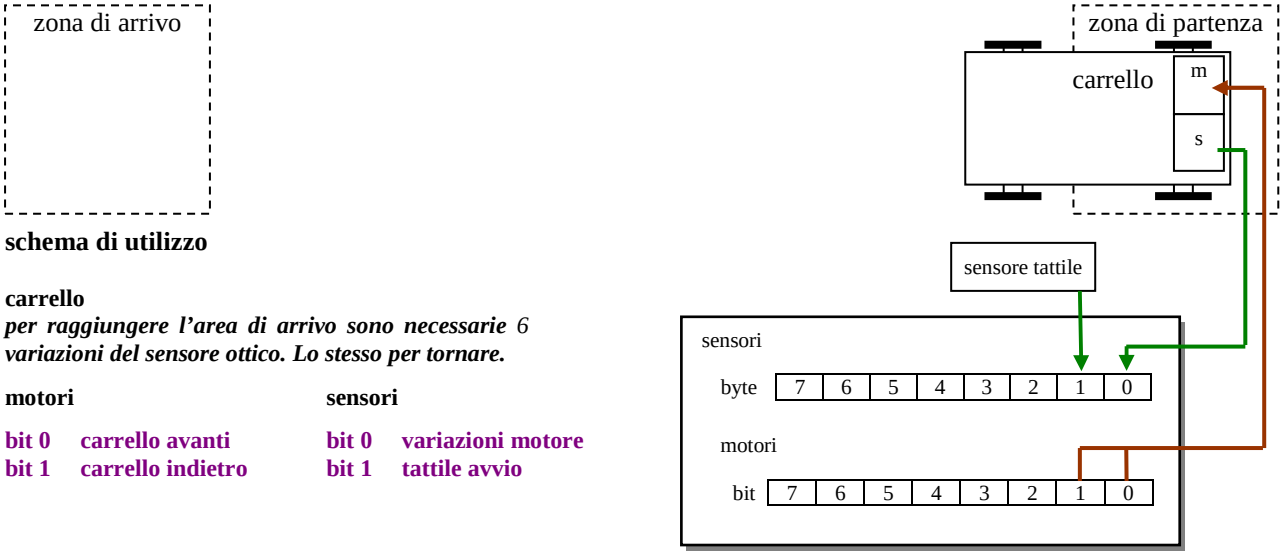
```
Progetto1 - MIOS_Simple (codice)
connetti                                    Click
Private Sub connetti_Click()
    Call nconnect(13)
    Call nwrite(9, 1)
    Call muovi(0, 0, 4)
    Call ndisconnect
    ok.Text = "ok"
End Sub

Private Sub esci_Click()
    End
End Sub

Private Sub Form_Paint()
Print "prova con valori fissi / porta USB n°13"
CurrentY = 2500
Print "variazioni ";
End Sub
```





Rivediamo i problemi già affrontati in Liberty Basic:
(vedi negli esempi il programma ***carrello***).

Abbiamo un carrello con un motore collegato alle porte 0 (avanti) e 1 (indietro) e con un sensore ottico (porta 0) che ne conta le variazioni. Alla pressione di un sensore tattile (porta 1) si avvia verso la zona di arrivo. Qui effettuerà una sosta di 5 secondi per poi tornare alla zona di partenza.



schema di utilizzo

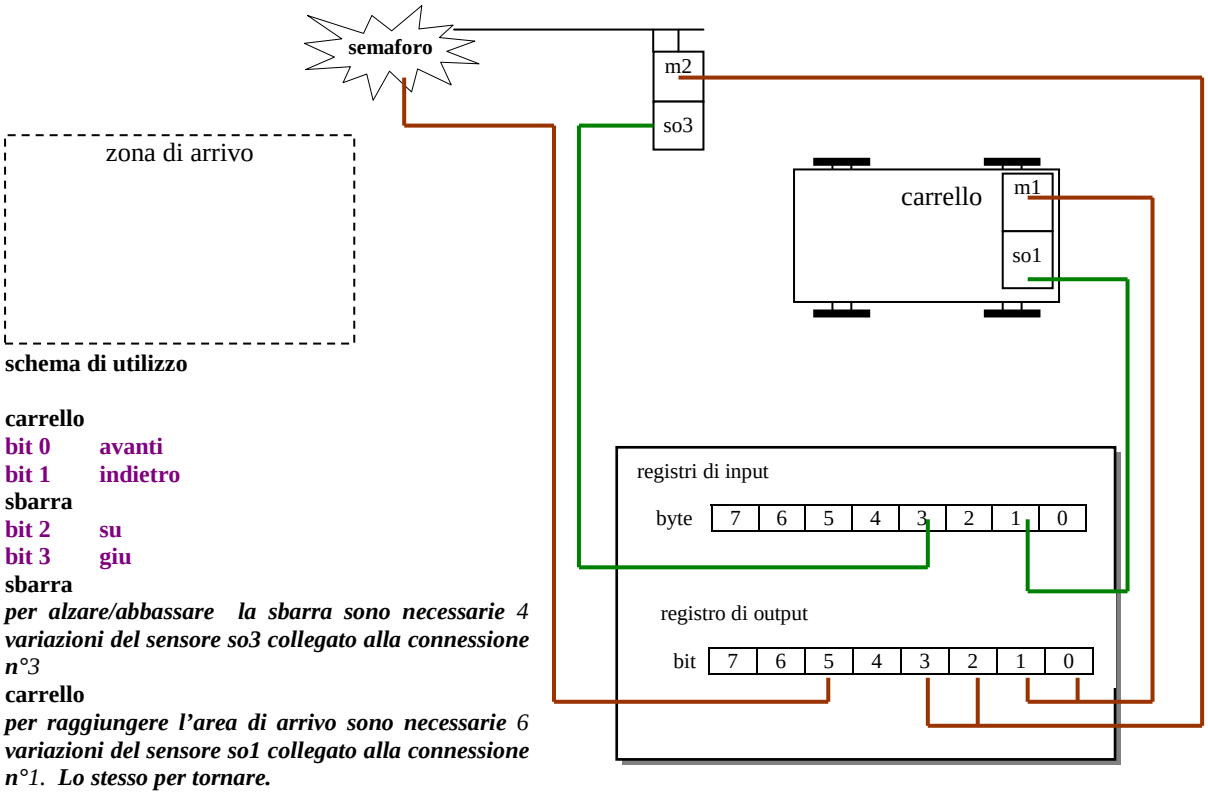
carrello
per raggiungere l'area di arrivo sono necessarie** 6 **variazioni del sensore ottico. Lo stesso per tornare.

motori		**sensori**	
bit 0	**carrello avanti**	**bit 0**	**variazioni motore**
bit 1	**carrello indietro**	**bit 1**	**tattile avvio**

Avendo già inserito i moduli gestione.bas e MZ.bas scrivi su un foglio a parte il codice del programma.

2° problema

*All'avvio del programma si accende il semaforo (**bit 5**) e inizia ad abbassarsi il passaggio a livello (motore **m2** con sensore contagiri **so3**). Quando si è abbassato parte il carrello (motore **m1** con sensore contagiri **so1**) e raggiunge la zona di arrivo. Qui si ferma 4 secondi per poi tornare alla zona di partenza. Quando è tornato si rialza il passaggio a livello e si spegne il semaforo.*



schema di utilizzo

carrello
bit 0 avanti
bit 1 indietro
sbarra
bit 2 su
bit 3 giu
sbarra
***per alzare/abbassare la sbarra sono necessarie** 4 **variazioni del sensore so3 collegato alla connessione n°**3*
carrello
per raggiungere l'area di arrivo sono necessarie** 6 **variazioni del sensore so1 collegato alla connessione n°**1. **Lo stesso per tornare.

Avendo già inserito i moduli gestione.bas e MZ.bas scrivi su un foglio a parte il codice del programma.

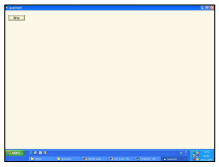
GEOMETRIA CON IL CALCOLATORE

VB, valorizzando al meglio le potenzialità della scheda grafica del calcolatore, permette la gestione dei singoli pixel del monitor (o, meglio, dei singoli twip) in modo da ottenere ottime prestazioni, non solo nella gestione delle immagini, ma anche nella realizzazione di vari tipi di grafici.

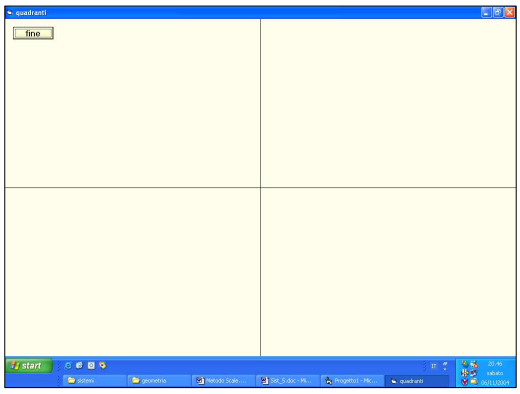
❑ Organizzazione dello schermo

Lo schermo (o form) a nostra disposizione può essere riorganizzato in modo da essere suddiviso nella quantità di spazi (twip) necessaria per l'attività da svolgere. Inoltre, la possibilità di assegnare valori negativi alle coordinate degli spazi, permette di suddividere la form in quattro quadranti cartesiani e di svolgere su di essa attività di disegno geometrico.
Al caricamento della form va dato l'ordine:

Scale(x1, y1)–(x2, y2)
nel quale x1 e y1 sono le coordinate date al vertice in alto a sinistra mentre x2 e y2 sono le coordinate date al vertice in basso a destra.



A destra la form ricavata con l'ordine:
Scale (0, 0)-(400, 400)
Un *CommandButton* permette di uscire dal programma (vedi *quadranti1*)



Segue la form ricavata con l'ordine:
Scale (-100,100)-(100,-100)
seguita degli ordini:
Line (-100,0)-(100,0)
Line (0,-100)-(0,100)
per evidenziale le linee di divisione tra i vari quadranti (vedi *quadranti2*).

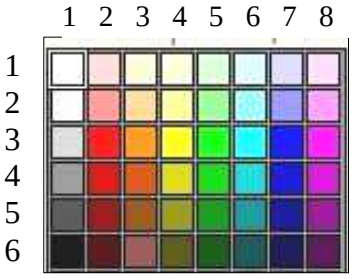
❑ La gestione dei colori

Tutti gli ordini grafici prevedono la possibilità di colorare in modo autonomo le singole linee. Contrariamente a quanto visto in GWBASIC, i codici dei colori non sono espressi in decimale ma in esadecimale (base sedici). La sintassi è la seguente:

&HBBVVRR&

dove la &(e commerciale) racchiude il codice e la H indica si sta utilizzando un numero esadecimale. BB, VV e RR indicano, in esadecimale, la quantità di blu, di verde e di rosso. Il valore può andare da 00 a FF.
Ad esempio il codice: *&HFF0000&* indica una linea di colore blu mentre *&HFF00FF&*indica un colore viola. Il valore esadecimale può essere gestito da una variabile numerica con le stesse modalità del valore decimale. In entrambi i casi il numero viene subito convertito nel corrispondente valore binario.
Ad esempio con *a = &HFF0000&* viene memorizzato dentro una variabile il valore del colore blu. I valori esadecimali dei colori potranno anche essere ricopiate dalla finestra delle proprietà oppure progettati dal programmatore.

Creazione di un archivio colori

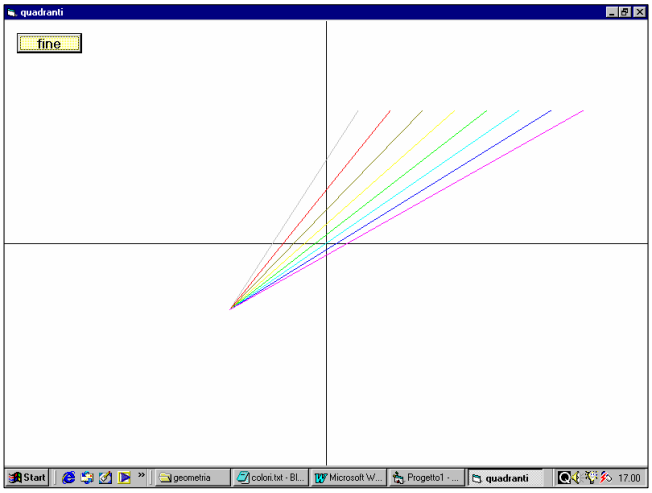


Se si vogliono disegnare più oggetti di diverso colore può essere utile organizzare un archivio colori prevedendo la sua gestione con vettore o matrice. La soluzione più funzionale è organizzare l'archivio con blocco note trasferendone poi il contenuto in RAM ogni volta che serve.
La seguente tabella riporta i codici della tavolozza colori del Visual Basic. Memorizzata sequenzialmente nel file colori.txt, può essere caricata in una matrice e utilizzata all'interno di un programma.

&HFFFFFF&	&HC0C0FF&	&HC0E0FF&	&HC0FFFF&	&HC0FFC0&	&HFFFFC0&	&HFFC0C0&	&HFFC0FF&
&HE0E0E0&	&H8080FF&	&H80C0FF&	&H80FFFF&	&H80FF80&	&HFFFF80&	&HFF8080&	&HFF80FF&
&HC0C0C0&	&H0000FF&	&H0080FF&	&H00FFFF&	&H00FF00&	&HFFFF00&	&HFF0000&	&HFF00FF&
&H808080&	&H0000C0&	&H0040C0&	&H00C0C0&	&H00C000&	&HC0C000&	&HC00000&	&HC000C0&
&H404040&	&H000080&	&H004080&	&H008080&	&H008000&	&H808000&	&H800000&	&H800080&
&H000000&	&H000040&	&H404080&	&H004040&	&H004000&	&H404000&	&H400000&	&H400040&

Con il seguente programma (quadranti_linee2.vbp) viene gestita la form collocata al fianco:

```
Dim d(6, 8) ' matrice numerica da utilizzare per memorizzare i colori da colori.txt
--------------------------------
Private Sub fine_Click()
End
End Sub
--------------------------------
Private Sub Form_Paint()
'disegno dei quadranti cartesiani
Call colori
Scale (-100, 100)-(100, -100)
Line (-100, 0)-(100, 0)
Line (0, -100)-(0, 100)
For kk = 1 To 8
Line (-30, -30)-(kk * 10, 60), d(3, kk)
Next kk
End Sub
--------------------------------
'lettura del file colori.txt per memorizzare i valori dei colori
'dentro la matrice d
Public Sub colori()
Open "colori.txt" For Input As 1
For k = 1 To 6
For kk = 1 To 8
Input #1, d(k, kk)
Next kk
Next k
End Sub
```



- **Gli ordini grafici**

Come abbiamo già visto, per disegnare delle **linee** si utilizza l'ordine:

Line (x1, y1)–(x2, y2)

dove x1, y1 e x2, y2 indicano le coordinate dei punti di inizio e fine. E' possibile aggiungere, dopo una virgola, il valore del colore. Ad esempio con:

Line (-100, 0)-(100, 0), &HFFFF00

Il calcolatore disegnerà una linea celeste.

Per disegnare un **punto** si utilizza:

Pset (x, y)

dove x, y ne indicano le coordinate. E' possibile aggiungere, dopo una virgola, il valore del colore.

Il disegno di **cerchi** può avvenire con l'ordine

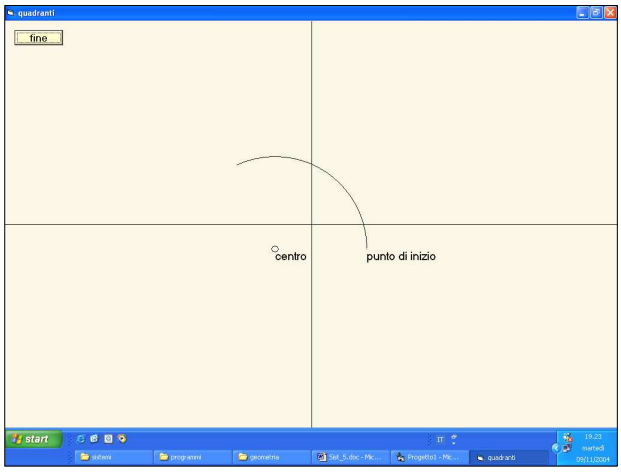
Circle (x, y), *raggio, colore, inizio, fine , aspetto*

x, y	Obbligatoria. Valori che indicano le coordinate del centro del cerchio, dell'ellisse o dell'arco.
raggio	Obbligatoria. Valore che indica il raggio della circonferenza, dell'ellisse o dell'arco.
colore	Facoltativa. Valore intero che indica il colore RGB del bordo del cerchio. Se omesso, viene utilizzato il valore della proprietà **ForeColor**.
inizio, fine	Facoltative. Quando si disegna un arco o una sezione di cerchio o di ellisse, *inizio e fine* specificano la posizione iniziale e la posizione finale dell'arco espressa in radianti. L'intervallo ammesso per entrambi gli argomenti è compreso tra -2 π e 2 π radianti. Il valore predefinito per *inizio* è 0 radianti. Il valore predefinito per *fine* è 2 π radianti.
aspetto	Facoltativa. Valore a precisione singola che indica le proporzioni del cerchio. Il valore predefinito è 1,0. Tale valore restituisce su qualsiasi schermo un cerchio perfetto (non ellittico).

inizio e *fine* permettono il disegno di archi. Il disegno avviene a partire da 0 (che è collocato a destra del centro) e procede in senso antiorario.

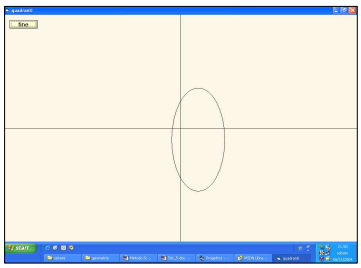
La form a fianco è ottenuta grazie all'esecuzione della seguente procedura:

```
Private Sub Form_Paint()
Scale (-100, -100)-(100, 100)
Line (-100, 0)-(100, 0)
Line (0, -100)-(0, 100)
Circle (-12, -12), 1
Print "centro"
PSet (18, -12)
Print "punto di inizio"
Circle (-12, -12), 30, , 0, 2
End Sub
```

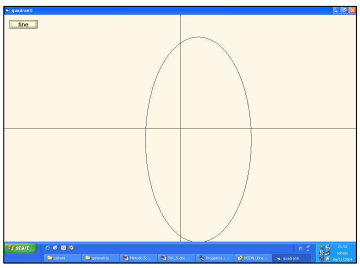


L'aspetto serve per disegnare le elissi e indica il rapporto tra raggio orizzontale e verticale. Se il valore è 1 si ha un cerchio. Con un valore positivo, *raggio* è applicato al raggio verticale mentre il valore di *aspetto* indica di quante volte il raggio orizzontale è ridotto. Con un valore negativo, *raggio* è applicato al raggio orizzontale mentre il valore di *aspetto* indica di quante volte il raggio verticale è aumentato.

Circle (10, -10), 30, , , , 2



Circle (10, -10), 30, , , , -2

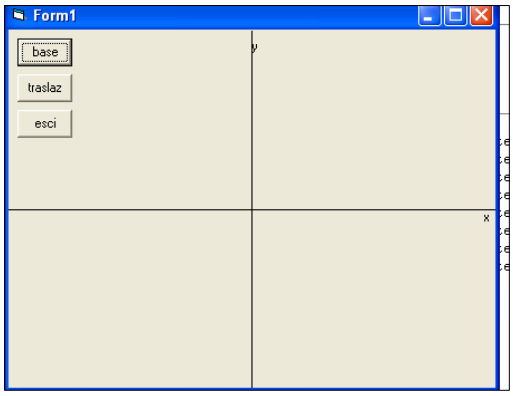


❑ Il compito di esame del 2008

Vediamo ora il compito che à stato dato agli esami di Stato del 2008. Nella prova di matematica i candidati dovevano eseguire su un foglio a quadretti una traslazione di una figura i cui dati erano fissi.
Nella prova di informatica, utilizzando gli stessi dati come suggeriti, bisognava realizzare il software di gestione in Visual Basic.

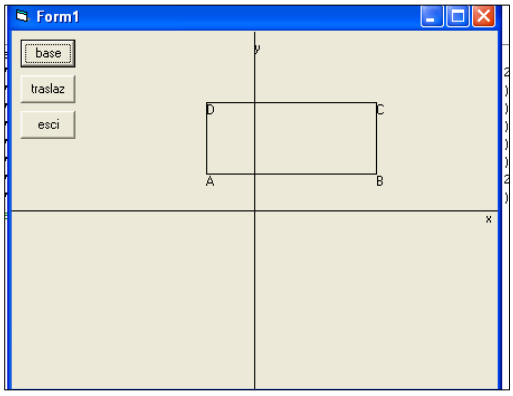
la finestra del codice >

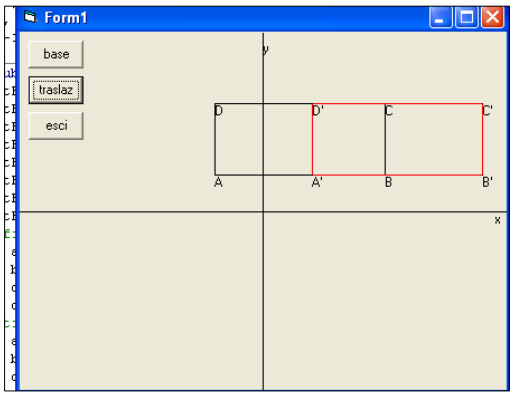
```
Public ax, ay, bx, by, cx, cy, dx, dy As Integer
Private Sub Form_Click()
Line (-4, -4)-(0, -4)
End Sub
Private Sub Form_Paint()
Scale (-10, 10)-(10, -10)
Line (-10, 0)-(10, 0): PSet (0, 9.5): Print "y"
Line (0, -10)-(0, 10): PSet (9.5, 0): Print "x"
End Sub
Private Sub base_Click()
ax = InputBox("scrivi la coordinata x di A", "inserisci coordinate", -2)
ay = InputBox("scrivi la coordinata y di A", "inserisci coordinate", 2)
bx = InputBox("scrivi la coordinata x di B", "inserisci coordinate", 5)
by = InputBox("scrivi la coordinata y di B", "inserisci coordinate", 2)
cx = InputBox("scrivi la coordinata x di C", "inserisci coordinate", 5)
cy = InputBox("scrivi la coordinata y di C", "inserisci coordinate", 6)
dx = InputBox("scrivi la coordinata x di D", "inserisci coordinate", -2)
dy = InputBox("scrivi la coordinata y di D", "inserisci coordinate", 6)
'disegno figura
Line (ax, ay)-(bx, by)
Line (bx, by)-(cx, cy)
Line (cx, cy)-(dx, dy)
Line (dx, dy)-(ax, ay)
'nomi punti
PSet (ax, ay): Print "A"
PSet (bx, by): Print "B"
PSet (cx, cy): Print "C"
PSet (dx, dy): Print "D"
End Sub
Private Sub trasl_Click()
'disegno figura
Line (ax + 4, ay)-(bx + 4, by), &HFF&
Line (bx + 4, by)-(cx + 4, cy), &HFF&
Line (cx + 4, cy)-(dx + 4, dy), &HFF&
Line (dx + 4, dy)-(ax + 4, ay), &HFF&
'nomi punti
PSet (ax + 4, ay): Print "A'"
PSet (bx + 4, by): Print "B'"
PSet (cx + 4, cy): Print "C'"
PSet (dx + 4, dy): Print "D'"
End Sub
Private Sub esci_Click()
End
End Sub
```



< la form all'inizio

la figura di base >





< la figura traslata