CORSO PRATICO DI VISUAL BASIC

RICHIAMI DI PROGRAMMAZIONE

Per **programmazione** si intende il procedimento di stesura della sequenza di istruzioni e comandi, scritti secondo le regole di un certo linguaggio per elaboratori elettronici.

Per **programma** si intende una serie di un insieme di istruzioni e comandi per svolgere in modo chiaro e non ambiguo una determinata funzione.

Un **algoritmo** non è altro che una procedura che tenta di risolvere un problema applicando un certo numero di passi.

Risolvere un problema significa:

- trovare un procedimento che consenta di produrre i risultati, a partire dai dati iniziali, attraverso un processo di elaborazione.

La metodologia di lavoro prevede due fasi:

- ✓ **Analisi del problema.** In questa fase occorre:
  a. definire con precisione i dati che abbiamo a disposizione sui quali basare la soluzione del problema detti dati di ingresso (**INPUT**);
  b. le soluzioni adottate, ovvero il procedimento che permette di passare dai dati iniziali ai risultati (**ELABORAZIONE**);
  c. si devono definire i risultati attesi, detti anche dati di uscita (**OUTPUT**).

- ✓ **Stesura dell'algoritmo**. La soluzione del problema che scaturisce dall'analisi del problema deve essere organizzata e sviluppata in una serie di operazioni da attuare in un ordine ben definito, per ottenere i risultati attesi e in questo consiste la stesura dell'algoritmo.
- ✓ **Algoritmo**: descrizione di una serie finita di istruzioni che devono essere eseguite per raggiungere il risultato stabilito in precedenza.

Sezione dichiarativa: costanti e variabili

I dati che vengono definiti possono essere:

COSTANTI

- ✓ Dichiarare una costante significa associare un simbolo ad un valore.
- ✓ A differenza delle variabili tale valore non cambierà mai durante tutto il programma
- ✓ In ogni caso una costante NON occupa alcuna zona di memoria.
- ✓ Per dichiarare una costante ci sono due tecniche:

Const nome=Valore oppure **Const nome as Tipo=Valore**
const pi = 3,1415 oppure **Const pi as Double=3,1415**

VARIABILI

In ogni linguaggio di programmazione, ci troveremo a gestire una classe variegata di informazioni:

1. numeri
2. stringhe, ossia dati alfanumerici (composto da numeri e lettere);
3. data e ora
4. informazioni booleane (di tipo True oppure False)

5. tipi di dati definiti dall'utente
6. tipi di dati strutturati (array monodimensionali e pluridimensionali) Gli array pluridimensionali sono le matrici. Gli array ad una sola dimensione sono detti anche vettori.

Per gestire questo mixing di informazioni, nella creazione di programmi si richiede **la memorizzazione dei valori** in variabili.

- ✓ Una variabile è un'area di memorizzazione temporanea.
- ✓ Una variabile cambierà durante l'esecuzione del programma.
- ✓ Per dichiarare una variabile si utilizza **l'istruzione Dim:**

Dim Nome as Tipo

TIPI DI DATO

TIPO	DATI	OCCUPAZIONE DI MEMORIA	INTERVALLO DI VALORI
BYTE	NUMERI INTERI	1 BYTE	DA 0 A 255 → $2^8 = 256$
INTEGER	NUMERI INTERI	2 BYTE	DA -32768 A 32767
LONG	NUMERI INTERI	4 BYTE	CIRCA +/- 2 MILIARDI
SINGLE	NUMERI DECIMALI A SINGOLA PRECISIONE.	4 BYTE	DA +/- 1,401298E-45 A 3,402823E48
DOUBLE	NUMERI DECIMALI A DOPPIA PRECISIONE	8 BYTE	DA +/- 4,94065645841247E-324 A 1,79769313486232E308
CURRENCY	NUMERI IN FORMATO VALUTA (4 DECIMALI)	8 BYTE	+/- 9E14
STRING	MEMORIZZA STRINGHE ALFANUMERICHE	1 BYTE	INSIEME DA 0 A 255 CARATTERI
BOOLEAN	VALORI LOGICI	2 BYTE	VERO O FALSO
DATE	DATE E ORE	8 BYTE	DA 1/1/100 AL 12/31/9999
VARIANT	UNO QUALSIASI DEI DATI PRECEDENTI		

ASSEGNAZIONE DI UNA VARIABILE:

per scrivere un valore dentro una variabile si usa l'operatore di assegnazione, che è rappresentato dal simbolo =.

Quindi, se scrivo **a = 12** assegno alla variabile a il valore numerico 12.

STESURA DELL'ALGORITMO

individuate le parti che costituiscono la soluzione del problema, occorre descrivere la tecnica con la quale si intende realizzare ciascuna di queste parti. Le tecniche possono essere:

a. *Lo pseudocodice o linguaggio di progetto*: è una descrizione verbale in linguaggio corrente delle istruzioni del problema;

b. *Flow chart o diagramma di flusso*: è una rappresentazione simbolica o grafica delle azioni da compiere.

Esempio: Calcolo dell'importo totale di una fattura con applicazione dell'IVA.

Variabili	Input	Output	Lavoro
Imponibile	X		
IVA	X		
Imposta			X
TotaleFattura		X	

Algoritmo in pseudocodice o in linguaggio di progetto:

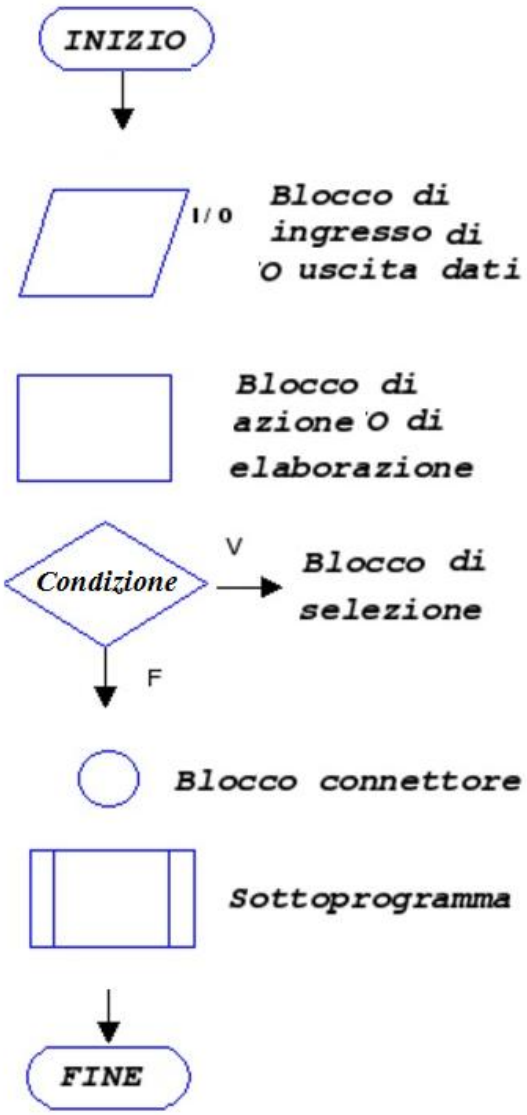
INIZIO
Leggi l'imponibile
Leggi l'aliquota IVA
Calcola l'imposta
Calcola il Totale Fattura
Scrivi il Totale Fattura
FINE

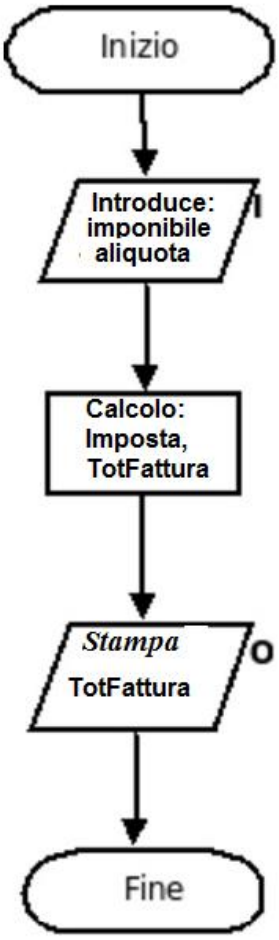
Tabella di traccia dell'algoritmo

N° istruzioni	Imponibile	Aliquota	Imposta	TotFattura	Input	Output
1						
2	500				Lettura dati	
3		20%			Lettura dati	
4			100			
5				600		
6						600
7						

Algoritmo in flow chart o diagramma di flusso:

permette di descrivere gli algoritmi mediante simboli grafici contenenti le operazioni da eseguire.





PROGRAMMAZIONE STRUTTURATA

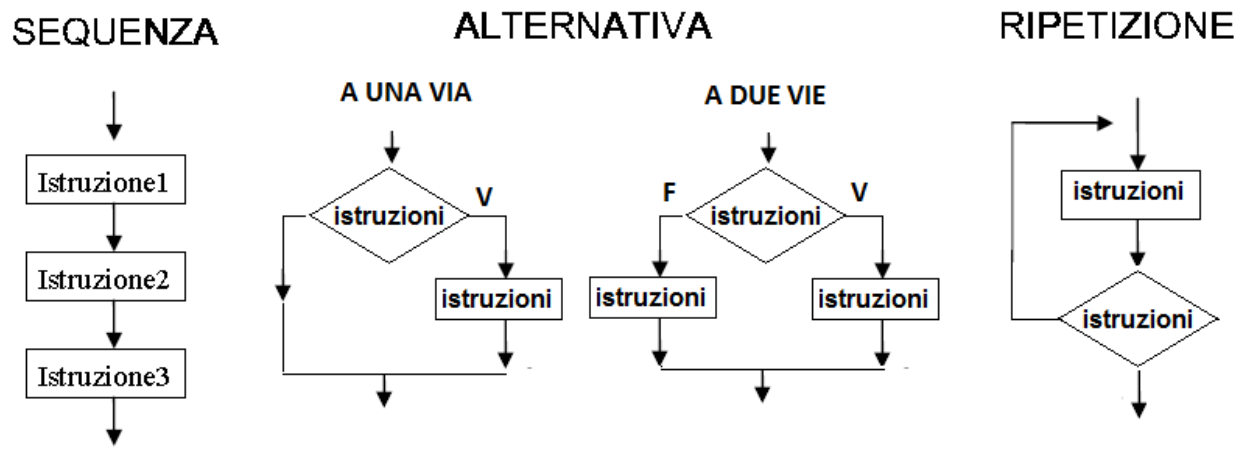
Il principio delle strutture di programmazione si basa sul teorema di **Bohm-Jacopini**, nel quale si afferma che qualunque algoritmo rappresentabile mediante flow-chart, può essere trasformato in un insieme costituito solamente da tre tipi di strutture elementari, che sono:

- **Sequenza**;
- **Selezione, scelta o alternativa**;
- **Ripetizione, iterazione** o **ciclo**.

Il pseudocodice dei tre modelli

SEQUENZA	ALTERNATIVA	RIPETIZIONE
…….	SE condizione ALLORA	ESEGUI
Istruzione1	Istrizione-a	Istruzioni
Istruzione2	ALRIMENTI	RIPETI FINCHE' condizione
Istruzione3	Istruzione-b	
……	FINE SE	

Diagramma a blocchi (flow chart)



CONTATORI E TOTALIZZATORI

1) Contare: si realizza con un contatore che è un tipo di variabile numerica cui si assegna un *valore iniziale = 0* (azzeramento della variabile) e si incrementa di 1 ogni volta che si verificano le condizioni poste.
L'espressione usata per ***incrementare il contatore*** è:

CONTATORE = CONTATORE + 1

2) Totalizzare: operazione analoga al calcolo della **TotFattura** del esempio precedente. Oppure paragonato alla spesa effettuata da un cliente in un supermercato dove i prezzi dei prodotti vengono sommati e alla fine si ottiene il totale.
L'operazione si realizza con un totalizzatore che è una variabile di tipo numerico cui si assegna un valore iniziale = 0 (azzeramento della variabile) incrementando di un valore numerico il totalizzatore ogni volta che si verificano le condizioni poste. L'espressione usata per incrementare il totalizzatore è:

TOTALE = TOTALE + NUMERO

Il COMPUTER E L'ALGORITMO

L'algoritmo, per essere eseguito da un computer deve essere tradotto in un linguaggio macchina (0, 1) di basso livello e sia compilato da un linguaggio di programmazione.

L'algoritmo così tradotto prende il nome di programma e la persona che è in grado di scriverlo si chiama programmatore.

Le fasi successive all'analisi del problema e alla diagrammazioni sono quindi:

Codifica oppure codice sorgente : scrittura del programma nel linguaggio di programmazione scelto che può essere digitato direttamente nel sistema.
Correzione: (Debugging degli errori) consiste nella prova del programma e nella correzione di eventuali errori di sintassi o logici.
il **codice oggetto** è la traduzione del codice sorgente in linguaggio macchina, comprensibile solo all'elaboratore;

Manutenzione: per mutate esigenze potrebbe essere necessario, a distanza di tempo, la modifica del programma.
Il **linguaggio macchina** è un linguaggio di basso livello in cui sono scritti i programmi eseguibili per computer. È basato dunque sul codice *binario.*

VISUAL BASIC.net

E' un linguaggio di **programmazione visuale**, il che vuol dire che l'attività di sviluppo del software è **basato** sull'uso **dell'interfaccia grafica** (finestre, pulsanti, icone …)
Visual basic è un linguaggio di programmazione **orientati** agli **oggetti** (buttoni, etichette, caselle di testo, …) e agli **eventi**. (sono le azioni provocate dall'utente sugli oggetti : click, apertura del form ecc..).

L'AMBIENTE DI LAVORO DI VISUAL BASIC

Apriamo, ufficialmente per la prima volta, l'ambiente di lavoro Visual Basic. Dalla barra delle applicazioni di **Avvio**, apriamo la cartella **Programmi** e quindi **Microsoft Visual Basic 6.0**.

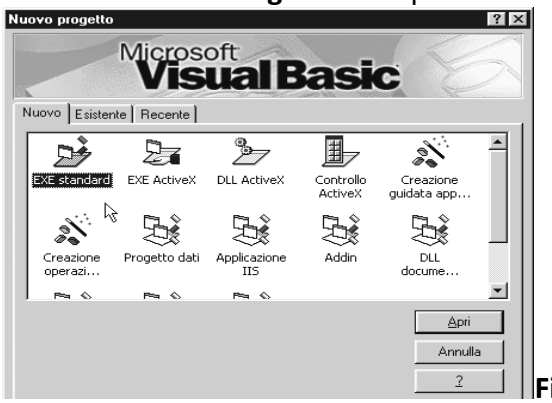


Figura 1 - La finestra di apertura di Visual Basic.

Appare una Finestra la cui caption (titolo o sottotitolo), nella riga blu in alto, è **Nuovo Progetto**
Ci viene chiesto quale tipo di progetto abbiamo in mente di realizzare e, in particolare, se si tratta:
1. di un progetto **Nuovo**, 2. di un progetto già **Esistente** oppure 3. di un progetto utilizzato di **Recente**.

Tralasciamo tutte le opzioni che compaiono nella finestra e fermiamoci alla prima icona, che porta l'indicazione **EXE standard**. L'opzione EXE standard è di uso comune.
L'ambiente di lavoro Visual Basic si presenta piuttosto affollato (figura 2).
In termini tecnici, siamo di fronte ad un MDI (Multiple Documents Interface, interfaccia a documenti multipli32) costituito da:
- la Finestra del form (il rettangolo grigio con una griglia interna a puntini: questo è l'unico l'elemento che apparirà all'utente in fase di esecuzione del programma)
- la Finestra del **Progetto**
- la Finestra **Disposizione Form**.

Casella degli strumenti: in cui gli oggetti dell'interfaccia grafica. vengono chiamati controlli.

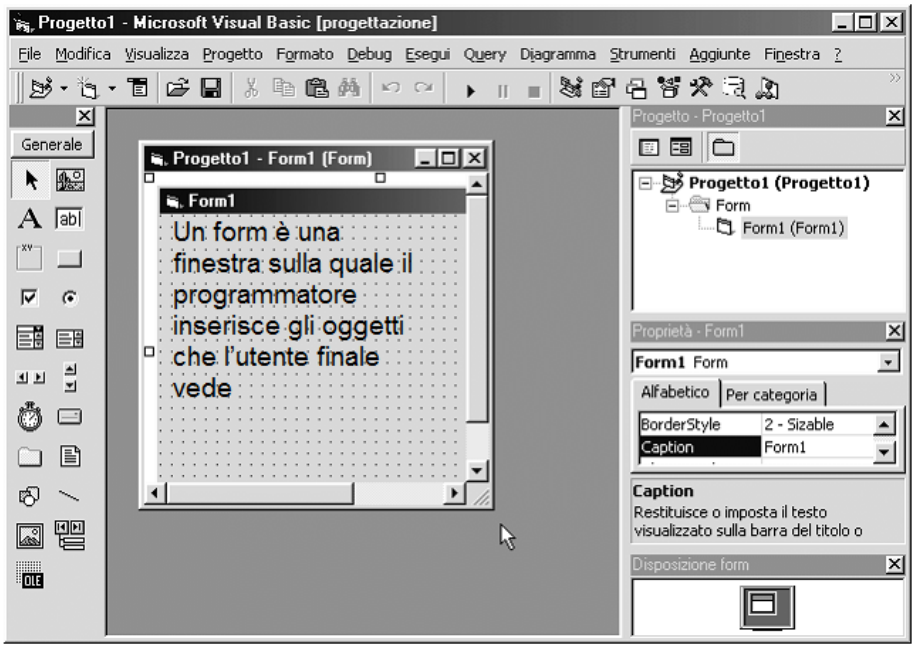


Figura 2

La programmazione VB si basa su alcuni concetti fondamentali:

Oggetti: gli oggetti per la creazione dell'interfaccia grafica sono i **form** (finestra, piattaforma) e i controlli (strumenti disegnati dal programmatore sul form).
Proprietà: sono le caratteristiche degli oggetti (nome, colore, dimensione ecc..).
Eventi: sono le azioni provocate dall'utente sugli oggetti (click, apertura del form ecc..).

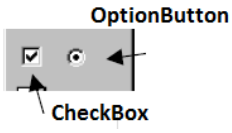
Codice: per far eseguire delle azioni al programma andiamo ad associare il codice (istruzioni vb) agli eventi.
Metodi: Rappresentano le operazioni eseguibili da un oggetto (routine già presenti nel linguaggio VB associate agli oggetti).

LA CASELLA DEGLI STRUMENTI
L'interfaccia grafica di un programma si basa sui form, all'interno dei quali vengono collocati, secondo le esigenze e le intenzioni del programmatore, i controlli che si trovano nella Casella degli Strumenti
Gli altri 20 controlli standard possono essere raggruppati, a seconda delle loro funzioni,
nelle otto categorie indicate di seguito.

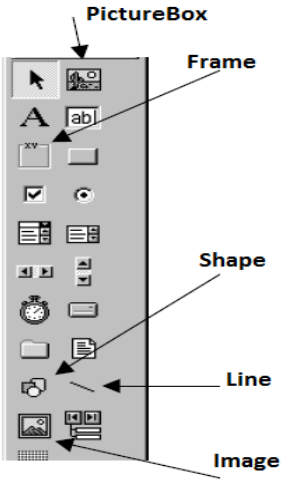
1. **Controlli che vengono utilizzati per lo scambio di informazioni con l'utente**, per comunicare all'utente dati o testi, o per ricevere dall'utente di dati o testi. Fanno parte di questa categoria 2 controlli:
1.1. **Label** (etichetta): permette di inserire testi che non possono essere modificati dall'utente, come ad esempio il risultato di una operazione, la didascalia di una foto, il titolo di un testo, una nota informativa...
1.2. **TextBox** (contenitore di testo): permette all'utente del programma di inserire dei dati, dei numeri, delle parole, dei testi. Questi dati vengono poi gestiti dal codice nello sviluppo del programma.

2. il controllo **CommandButton** (pulsante di comando) è un **controllo finalizzato a fare eseguire al programma determinate azioni/procedure/routine o macro.**

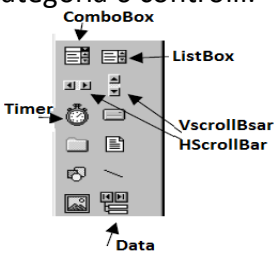
3. **Controlli di opzioni tra possibilità diverse, per fare procedere il programma secondo le scelte effettuate dall'utente**.
Fanno parte di questa categoria 2 controlli:
3.1. **OptionButton**: pulsante per la scelta di una sola opzione.
3.2. **CheckBox:** contenitore per la scelta una o più di opzioni



4. **Controlli per la gestione delle componenti grafiche inserite nel form.** Fanno parte di questa categoria 5 controlli:
4.1. **PictureBox** (contenitore di immagini): permette di visualizzare immagini grafiche.
4.2. **Frame** (cornice): viene utilizzato come contenitore per raggruppare i controlli. **Prima** si colloca il controllo Frame e **poi** al suo interno i controlli che vi si vogliono inserire.
4.3. **Shape** (forma): Consente di disegnare diverse forme geometriche nei form.
4.4. **Line** (linea): Disegna nel form linee con stili e grandezze diverse.
4.5. **Image** (immagine): Visualizza nel form un'immagine grafica. È simile al controllo Picturebox, ma svolge un numero ridotto di funzioni. In compenso utilizza una minore quantità di risorse di memoria, con il risultato di rendere più veloce il funzionamento di un programma.



5. **Controlli di dati per gestire elenchi o archivi di dati.** Fanno parte di questa categoria 6 controlli:



5.1. **ComboBox** (contenitore combinato, composto di un elenco e di un testo). L'utente può selezionare una voce dall'elenco o immettere una voce nuova scrivendola nel contenitore di testo.

5.2. **ListBox** (contenitore di un elenco di voci). Questo controllo viene utilizzato per visualizzare un elenco di voci tra le quali l'utente può sceglierne una.

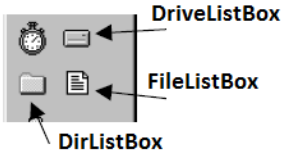
5.3. **HScrollBar** e **VscrollBsar** (barre di scorrimento orizzontale e verticale): forniscono due strumenti grafici per spostarsi in maniera rapida in un elenco di grandi dimensioni o in una notevole quantità di informazioni.

5.4. **Data** (dati): questo controllo consente di collegare un programma ai dati di un archivio (database).

6. Il **Timer** (temporizzatore), è il **controllo che misura il trascorrere del tempo.** Il controllo Timer viene collocato sui form come ogni altro controllo ed è visibile dal programmatore come una icona con l'immagine di un orologio. Rimane invece invisibile in fase di esecuzione e non può essere visto dall'utente del programma.

7. **Controlli per il collegamento con le unità di memoria** (disco fisso, floppy disk, CD-ROM). Fanno parte di questa categoria 3 controlli:

7.1. **DriveListBox** (contenitore della lista dei drive): mostra le unità disco che fanno parte del sistema.



7.2. **FileListBox** (contenitore della lista dei files): visualizza gli elenchi dei files contenuti nelle cartelle all'interno delle unità disco.

7.3. **DirListBox** (contenitore della lista delle cartelle): mostra le cartelle contenute nelle unità disco.

8. Il controllo OLE è l'unico controllo dell'ultima categoria di controlli: **serve a gestire il collegamento ad oggetti esterni** a Visual Basic. OLE sta per *Object Linking or Embedding*, cioè: collegamento e/o incorporazione di oggetti.

Consente di collegarsi ad altri programmi per incorporare oggetti esterni. Il controllo OLE visualizza e gestisce i dati da altre applicazioni per Windows, quali Microsoft Word e Excel

Esercitazione 1 (label e button)

Progettare un'applicazione che, al click del mouse su un button visualizzi un messaggio in una label "etichetta".

In questo progetto utilizzeremo due controlli: label (etichette) e button (pulsanti), il primo ha una duplice funzione, quella di descrivere il contenuto di altri controlli e quella di utput di dati. Il button viene detto anche pulsante di comando perché viene utilizzato per provocare l'esecuzione di istruzioni.

1- Apriamo un nuovo progetto assegnando il nome Esercizio1.
2- Nel progetto dell'interfaccia si prevede l'inserimento di due controlli. Disegniamo un button e una label
3- La terza fase consiste nell'assegnare le proprietà agli oggetti: Il *name* della label sarà **lblMessaggio**, Per il button il *name* sarà **btnMessaggio** e nella proprietà text o caption inseriremo il testo "Messaggio" che apparirà sul bottone .
   Cliccare sul buttone per inserire il seguente codice **lblMessaggio**.Caption = "La mia classe è favolosa"

Possiamo aggiungere due pulsanti: uno per chiudere l'applicazione, che chiameremo Esci e uno per cancellare il contenuto della label che chiameremo Cancella. Modifichiamo le proprietà dei due controlli:

Esci Name: btnEsci Caption: Esci

Cliccando sul bottone per inserire la funzione **end**

Cancella Name: btnCancella Caption: Cancella
Cliccando sul bottone per inserire l'assegnazione **lblMessaggio.caption = ""**

Modifica proprietà oggetto

WITH ... END WITH
Quando vogliamo valorizzare le proprietà di un oggetto, invece di scrivere una sequela di istruzioni come questa:

```
lblNome.caption = "Titolo"
lblNome.fontName = "Arial"
lblNome.fontSize = "12"
```

a volte può risultare più comodo usare questa sintassi, che fa esattamente la stessa cosa della precedente:

```
with lblNome
.caption = "Titolo"
.fontName = "Arial"
.fontSize = "12"
end with
```

Con nome ogetto
Dare un sottotitolo = "Titolo"
Tipo di carattere = "Arial"
Dimensione carattere = 12
Fine con

Il codice risulta più leggibile é eseguito più velocemente dal pc.

Esercitazione 2 (textbox)

Creare un progetto che, dato in input il proprio nome con casella di testo, lo visualizzi in un'altra casella di testo e in una label.
Inserire un commando esegui in cui viene inserito il codice e altri due bottoni ESCI e Cancella.

Esercitazione 3 (Calcoli)

Creare un progetto che, dati in input due numeri, visualizzi il risultato della loro somma.
Per effettuate calcoli in visual basic utilizziamo i seguenti operatori aritmetici.
+ somma
- sottrazione
*** moltiplicazione**
/ divisione
^ elevamento a potenza.
sqr(): una funzione che effettua la radice quadrata.
Val(): converte ad esempio il contenuto di una casella di testo in un valore numerico.
Inseriamo nel form due textbox per l'input dei due numeri, una label per visualizzare l'output e tre pulsanti: il primo per eseguire il calcolo e gli altri due per cancellare e uscire dal programma.
SetFocus : un metodo che posizione il cursore su un oggettto, nomeoggetto.metodo: txtbox1.setfocus, dopo il calcolo si riposizionare il cursore nella textbox del primo numero per effettuare un'altra addizione.

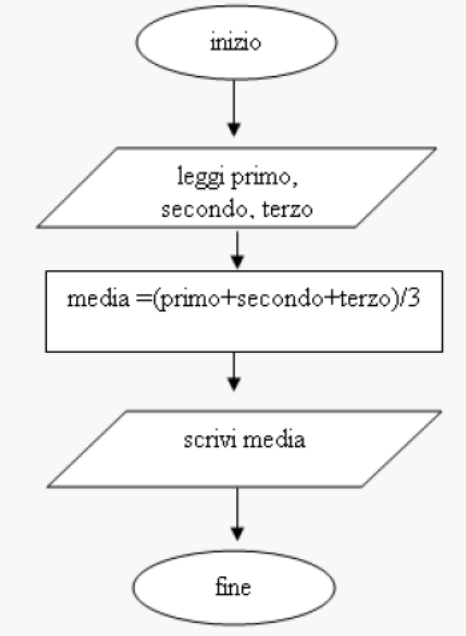
Esercitazione 4 (Struttura sequenziale)

Dati in input tre numeri calcolare la media aritmetica.
Vediamo quali sono le variabili da utilizzare:

VARIABILI			
	INPUT	OUTPUT	LAVORO
primo	X		
secondo	X		
terzo	X		
media		X	

Scrivere i passi da eseguire con un diagramma a blocchi:

creare l'interfaccia con tre caselle di testo di input e una etichetta per l'out put, un bottone per il calcolo e altri due per uscire e terminare.




```
Dim primo, secondo, terzo, media as double
primo = txtPrimo.Text
secondo = txtSecondo.Text
terzo = txtTerzo.Text
media = (primo + secondo + terzo) / 3
lblMedia.Text = media
```

Esercitazione 5: Struttura alternativa

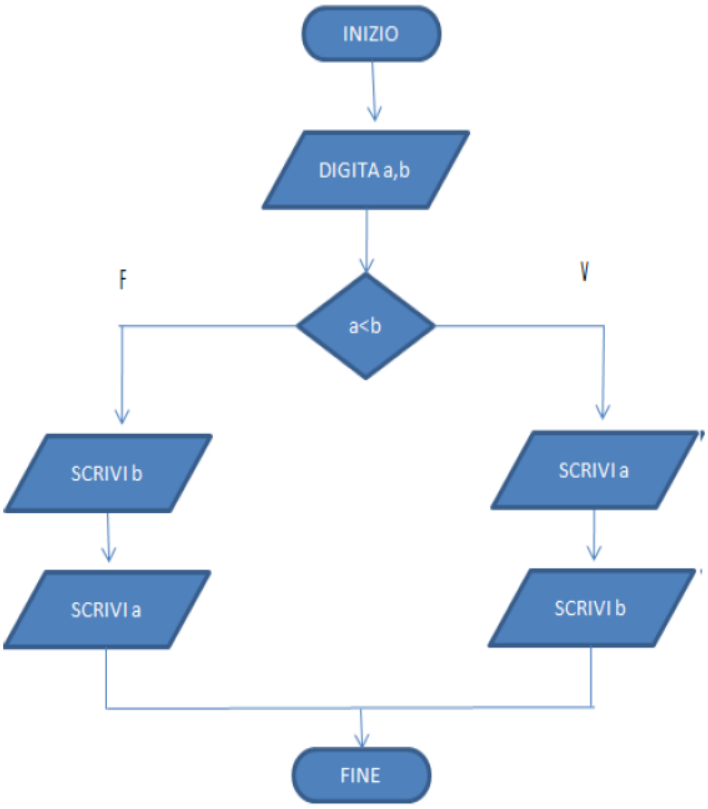
Pseudocodice	A una via	A due vie

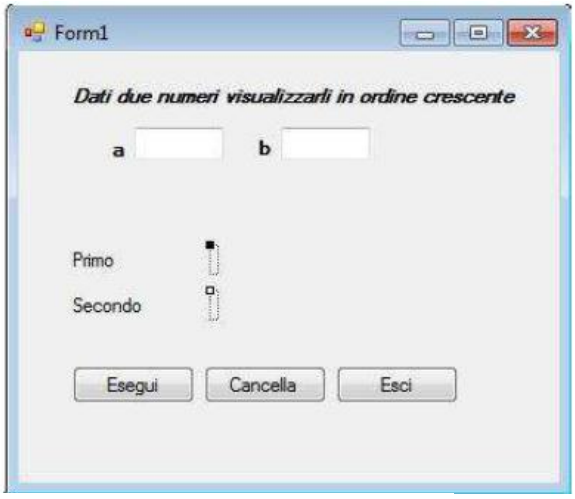
SE condizione
ALLORA
istruzione-a
ALTRIMENTI
istruzione-b
FINE SE
	if condizione then
	istruzione-a
	end if
		if condizione then
		istruzione-a
		else
		istruzione-b
		end if

Scrivere in ordine crescente due numeri dati in input.
DATI DI INPUT: due numeri interi a, b (utilizzare due caselle di testo)

DATI DI OUTPUT: i due numeri ordinati (utilizzare due label)
Tre bottoni : uno per il codice programma e due per uscire e cancellare.
Scrivere il pseudo codice del programma

Disegnare il flow chart






```
Private Sub btnEsegui_Click(ByVal sender As System.Object, ByVal
    Dim a, b As Integer
    a = txtA.Text
    b = txtB.Text
    If a < b Then
        lblPrimo.Text = a
        lblSecondo.Text = b
    Else
        lblPrimo.Text = b
        lblSecondo.Text = a
    End If
End Sub
```

```
Private Sub btnCancella_Click(ByVal sender As System.Object, ByV
    txtA.Text = ""
    txtB.Text = ""
    lblPrimo.Text = ""
    lblSecondo.Text = ""

End Sub
```

```
Private Sub btnEsci_Click(ByVal sender As System.Object, ByVal e
    End
End Sub
```

Nell'Esercizio 6: Costruiamo una calcolatrice, realizzeremo anche noi un programma simile; analizziamo quindi questo esempio soprattutto per vederne l'impostazione grafica e pensare a qualche miglioramento.

OPERATORI

Gli altri operatori sono di tre tipi: ***ARITMETICI***

Questi esempi possono essere provati scrivendoli direttamente nella Finestra Immediata (in questo caso non è necessario scrivere Debug.Print: è sufficiente il comando Print)

SIMBOLO	OPERAZIONE	SIMBOLO	OPERAZIONE
+	ADDIZIONE Esempio: Print 12 + 7 Risultato: 19	**+ o &**	CONCATENAMENTI DI STRINGHE "Salve " & "ragazzi" = Salve ragazzi
–	SOTTRAZIONE Esempio: Print 12 – 5 Risultato: 7	**^**	ELEVAMENTO A POTENZA Esempio: Print 4 ^ 2 Risultato: 16
*****	MOLTIPLICAZIONE Esempio: Print 12 * 5 Risultato: 60	**Sqr()**	RADICE QUADRATA Esempio: Print Sqr(16) Risultato: 4
/	DIVISIONE decimale Esempio: Print 100 / 8 Risultato: 12,5	****	DIVISIONE intera Esempio: Print 100 \ 8 Risultato: 12
Mod	RESTO della divisione Esempio: Print 100 Mod 8 Risultato: 4	**Int**	DIVISIONE intera (il risultato è uguale a quello che si ottiene con l'operatore \) Esempio: Print Int(100/8) Risultato: 12
Rnd	Genera numeri casuali "random" fra 0 e 0,9999	**Randomize Timer**	Insieme a Rnd permette di generare diversi numeri casuali

l'istruzione **Randomize** senza argomenti per inizializzare il generatore di numeri casuali con un valore iniziale basato sul timer di sistema.

Per generare numeri interi casuali compresi in un determinato intervallo, utilizzare la formula seguente:
Int((massimo - minimo + 1) * Rnd + minimo)

Dove *massimo* è il numero più alto nell'intervallo e minimoè il numero più basso nell'intervallo.

Esempio per generare due numeri casuali diversi fra 0 e 5, si risolve in questo modo:

```
Randomize Timer   'se non si utilizza Timer I numeri casuali alcune volte possono uscire uguali
a = Int(Rnd * 6)
  b = Int(Rnd * 6)
```

il risultato verrà visualizzato in una casella di testo

```
TextBox1.Text = a & " - " & b
```

Esercizio

Realizzare una specie di SLOT MACHINE, utilizzando:
4 caselle di testo (in una inserire il numero delle partite e nelle altre i numeri casuali riprodotti) e un label,
1 pulsante in cui viene inserito il codice.

Soluzione

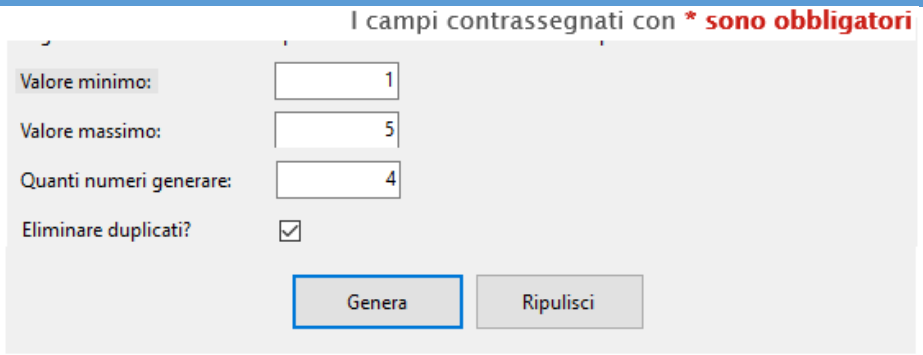
```
Dim i As Integer
 Dim LB As Integer
 Dim UB As Integer

Randomize Timer

 MINIMO = 1
 MASSIMO = 5
 TextBox1.Text = Int((MASSIMO) * Rnd + MINIMO)
 TextBox2.Text = Int((MASSIMO) * Rnd + MINIMO)
 TextBox3.Text = Int((MASSIMO) * Rnd + MINIMO)
 If TextBox1.Text = TextBox2.Text And TextBox2.Text = TextBox3.Text Then
 Label1.Caption = "VITTORIA!!"
 End If
 If Not (TextBox1.Text = TextBox2.Text = TextBox3.Text) Then
 Label1.Caption = "Ritenta!"
 End If
 a = TextBox4      ' numero partite
 If TextBox1.Text = TextBox2.Text And TextBox2.Text = TextBox3.Text Then
 Text4.Text = a + 15     ' incrementa di 15 le giocate
 End If
 a = TextBox4
 If Not (TextBox1.Text = TextBox2.Text = TextBox3.Text) Then
 TextBox4.Text = a - 1      ' decrementa le partite
 End If
 If TextBox4.Text = 0 Then
 Label1.Caption = "Game Over"
 End If
```

Esercizio

Si generano numeri *interi* compresi tra i valori minimo e massimo specificati nel modulo sottostante. Si deve creare una maschera che deve contenere (vedi figura)



3 caselle di testo, una checkbox, due pulsanti e box messaggi che deve contenere i numeri generati.

Il comando VbCrLf oppure vbLf

forza il computer ad andare a capo ed a scrivere il testo che segue all'inizio della riga successiva.

```
Dim Info As String
Info = "Per fare partire il programma premi il tasto F1."
Info = Info + vbCrLf + "CIAO CIAO"
Print Info
```

OPERATORI DI RELAZIONE		
= uguale	< minore	<= minore o uguale
> maggiore	<> diverso	>= maggiore o uguale

OPERATORI LOGICI	
AND	Prodotto logico
OR	Somma logica
NOT	Negazione

Esercitazione 7: Funzione InputBox:

Visualizza un messaggio in una finestra di dialogo, attende che l'utente immetta del testo o scelga un pulsante, quindi restituisce una stringa che include il contenuto della casella di testo.
Il formato intero della funzione inputBox() è il seguente:
varibilerisposta = InputBox(strPrompt [, strTitolo] [, strPredefinita] [, intPosX] [, intPosY]
dove:
strPrompt: è la domanda che deve esser posta dall'InputBox, ed è di tipo stringa
strTitolo: è il nome della nostra InputBox(), ed è di tipo stringa
strPredefinita: è la risposta predefinita che possiamo assegnare al campo dove l'utente andrà poi a digitare la propria risposta modificandola o lasciandola invariate, è sempre di tipo stringa.
IntPosX e **intPosY**: indicano la posizione iniziale in cui apparirà la finestra di input sul nostro schermo.

```
Dim risposta As String
risposta = InputBox("Qual'è la tua età?", "Immissione dell'età", "35", 7000, 3000)
Print risposta + "!"

nome = InputBox("Come ti chiami?", "Presentazione", "Scrivi qui il tuo nome")
Print "Ciao,"
Print nome + "!"
```

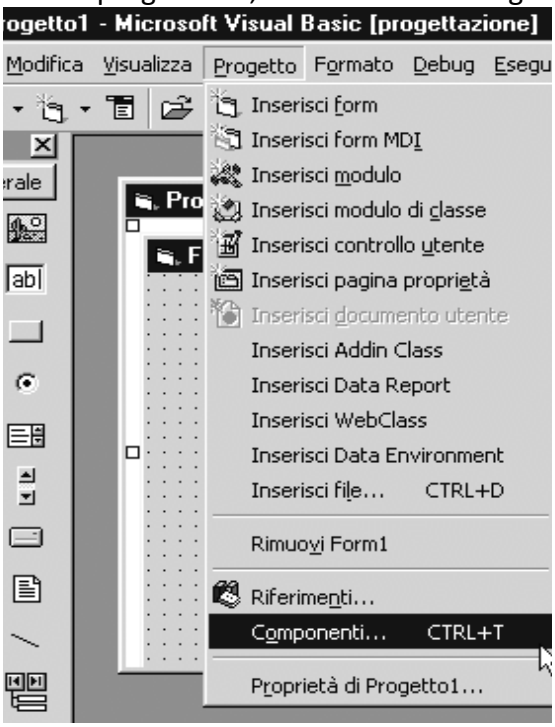
Se l'utente fa clic su Annulla, l'InputBox() restituisce una stringa di lunghezza zero Empty quindi per gestire la risposta è molto utile la seguente If:

```
if (risposta <> Empty) then
'codice che gestisce la risposta dell'utente diversa da una stringa vuota
Else
```

‘ codice che gestisce il clic sul pulsante annulla
End if

I CONTROLLI AGGIUNTIVI

Visual Basic mette a disposizione una ricca serie di controlli aggiuntivi che il programmatore può inserire in un suo programma, a seconda delle esigenze che gli si presentano.



Nella Finestra che si apre sono indicate tutte le serie di controlli aggiuntivi a disposizione del programmatore.
Facciamo tre clic con il tasto sinistro del mouse sulle tre caselle di queste serie di controlli aggiuntivi:

- **Microsoft Windows Common Controls 6.0**
- **Microsoft Windows Common Controls - 2 6.0**
- **Microsoft Windows Common Controls - 3 6.0**

come nella finestra che segue

Di questi controlli aggiuntivi, utilizzeremo nel manuale:

- il controllo **ImageList**, che consente di inserire in un programma una lista di immagini e di fare apparire le immagini nel form, richiamandole dalla lista, ogni volta che lo si desidera;
- il controllo **ProgressBar** (barra di avanzamento);
- il controllo **Slider** (dispositivo di avanzamento);
- il controllo **ToolBar** (barra di comandi);
- il controllo **Animation** (animazione grafica).

Utilizzeremo anche altri due controlli aggiuntivi che non sono compresi tra questi 15:

- il controllo Microsoft **MultiMedia** Control 6.0, che ci consentirà di inserire suoni, musiche e video-clip nei nostri programmi ;
- il controllo Microsoft **CommonDialog**, che utilizzeremo per gestire operazioni di salvataggio e di apertura di files.

Funzioni Matematiche:

Abs(X)	La funzione calcola il valore assoluto dell'argomento.
Cos(X)	La funzione calcola il coseno di un angolo espresso in radianti.
Exp(X)	La funzione calcola il valore della funzione esponenziale e^x.
Fix(X)	La funzione elimina la parte decimale di un numero reale.
Int(X)	La funzione calcola la parte intera di un numero reale.
Log(X)	La funzione calcola la parte intera di un numero reale.
Rnd(X)	La funzione restituisce un numero pseudocasuale compreso tra 0 e 1.

Round(espressione,numero di cifre decimali) : La funzione restituisce il valore arrotondato dopo il numero di cifre decimali specificato.

Sgn(X) La funzione restituisce -1 se x<0 , 0 se x=0, +1 se x>0 .

Sin(X) La funzione calcola il seno di un angolo espresso in radianti .

Sqr(X) La funzione calcola la radice quadrata di un numero reale.

Tan(X) La funzione calcola la tangente di un angolo espresso in radianti

Funzioni per la gestione delle Stringhe:

Asc(stringa) La funzione restituisce il codice ASCII del primo carattere della stringa specificata. Se l'argomento è una stringa vuota viene segnalato un errore di run- time.

Chr(codice) La funzione restituisce il carattere che ha il codice ASCII specificato.

LCase(stringa) La funzione converte le lettere maiuscole presenti nella stringa in minuscolo.

UCase(stringa) La funzione converte le lettere minuscole presenti nella stringa in maiuscolo.

Left(stringa, n) La funzione restituisce i primi n caratteri della stringa a partire da sinistra.

Len(stringa) La funzione restituisce il numero di caratteri che compongono la stringa.

LTrim(stringa) La funzione restituisce una stringa che rappresenta quella originale in cui sono stati eliminati spazi bianchi iniziali.

Format((dblSomma), "#,##0.00"): arrotonda in funzione alla normativa euro.

http://www.quesitiscienza.altervista.org/informatica/linguaggio_VisualBasic.html

http://www.webalice.it/kindofapple/oggetti.html