Dispensa di programmazione
Visual basic

Indice:

Scopo e argomenti del manuale
Questo manuale, con il corso ad esso associato, ha come scopo quello di fornire delle basi di programmazione utilizzando visual basic come esempio per creare programmi, utilizzare la programmazione ad oggetti grafici, concepire il programma modularmene

alekos.net

Cooperativa ALEKOS 20155 MILANO - V. Plana, 49 - P.IVA 11027820155

Tel 02 - 39264592 - Fax 02 – 700506084 info@alekos.net - www.alekos.net

Programma come file eseguibili e librerie: la scatola chiusa, compilatori e interpreti

Riprendiamo la stratificazione di un computer a partire dall'hardware (la parte fisica) fino ad arrivare all'utente, cioè al livello più vicino all'uomo (interfaccia e utilizzo).
Se l'hardware è il livello fisico, il software è il livello logico (in mezzo c'è il firmware cioè quelle informazioni e elaborazioni basilari che permettono al software di avviarsi, come ad esempio il BIOS).
Tutte le informazioni in un computer vengono registrate e lette (a livello software) in forma di file. Il file è quindi un insieme di byte che vengono letti (interpretati) diversamente a seconda del programma che li utilizza. Il sistema operativo stesso è fisicamente registrato in file.

E' quindi necessario riuscire a distinguere i file in varie categorie per comprendere se e come utilizzarli.
Il livello software si suddivide in sistema operativo, applicazioni e documenti utente.
Un programma è una scatola chiusa informatica che riceve un input (parametri), elabora e restituisce un output. Ad esempio il programma ping riceve il numero IP come parametro e restituisce il reply su schermo

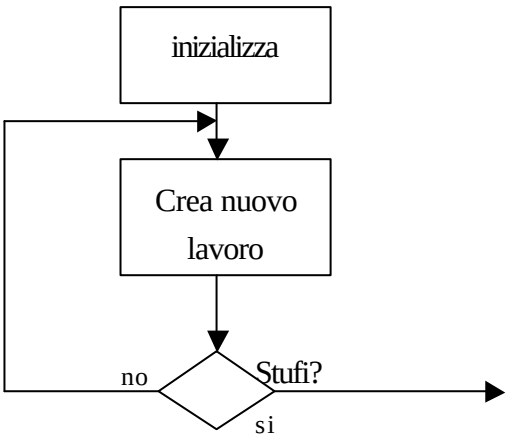
Ping 192.168.0.1

Con applicazione si intende l'insieme di programmi e file collegati che permettono al calcolatore di eseguire le funzioni che gli vengono richieste; mentre un sistema operativo è un insieme coerente di programmi, in grado di fare funzionare tutto l'HW.
Programmazione significa quindi creare i file (eseguibili .exe, librerie .dll, controlli activex .ocx) necessari per fare funzionare un programma.
Il processo di compilazione può quindi essere visto come un imbuto. Partendo dai sorgenti (file di codice, file di appoggio, file di progetto,..) il compilatore (programma solitamente compreso nell'ambiente di programmazione, come ad esempio l'ambiente visual studio, che comprende molti linguaggi, o più in particolare visual basic) crea questi file binari che sono il vero e proprio programma. Nel caso più semplice da un file di progetto (.vbp), dei file di form (.frm) e dei file modulo in codice visual basic (.bas) viene creato un file eseguibile (.exe).
Il codice nei sorgenti viene quindi scritto seguendo il linguaggio di programmazione del compilatore, per cui i sorgenti in C (o java o basic) verranno compilati con un compilatore C (o java o basic).
Un compilatore vero e proprio crea un eseguibile che può funzionare direttamente in altri computer (ovviamente con tutti gli altri file necessari al programma), cioè viene codificato in codice macchina.
Un interprete invece è un intermediario che traduce durante il funzionamento (runtime) il codice del programma in codice macchina. Ad esempio, in C esiste il compilatore, mentre il visual basic non crea veri eseguibili ma l'eseguibile ha codice che va ancora interpretato con il runtime (vbrun60.dll, ecc), il compilatore java crea classi (.class) che vanno interpretate dalle java virtual machine (JVM) che esistono nei vari sistemi operativi. L'idea dell'imbuto fa anche capire che si perdono informazioni, non si può infatti risalire al sorgente di un programma (a parte rari casi), anche perché la scrittura di un programma che in codice macchina realizza la stessa cosa può essere differente.

Concetto di algoritmo. Come funziona una macchina informatica. Il basic come linguaggio semplice per realizzare algoritmi

Il computer funziona più o meno come una macchina informatica. Un omino (il processore) dentro una scatola riceve ad ogni clock una riga di istruzioni in linguaggio macchina ed esegue l'istruzione utilizzando la piccola memoria a sua disposizione, operazioni semplici come addizione , spostamento in memoria,…

Con questa analogia si può comprendere il concetto di algoritmo, una serie di blocchi (azioni, domande,..) legati da un flusso di esecuzione, ovvero la sequenza di programma. I diagrammi a blocchi sono una ottima rappresentazione grafica di un algoritmo.



Inoltre il diagramma a blocchi rappresenta anche la modularità della programmazione, ogni blocco può essere dettagliato con altri algoritmi e così via, partendo dal generale per arrivare al particolare fino alle istruzioni scrivibili con un linguaggio di programmazione.

Ad esempio inizializza è azzerare una variabile (i), creare un nuovo lavoro è incrementare la variabile, la domanda stufi è la variabile è arrivata a 20?

In basic si può implementare in questo modo:

```
i=0
ricomincia:
i=i+1
if i>20 then go to ricomincia
```

O meglio senza utilizzare il desueto comando go to:

```
i=0
do while i<20
i=i+1
loop
```

Ci sono quindi vari modi per implementare lo stesso algoritmo, e dipende dallo stile di programmazione.

Istruzioni basic: subroutine, funzioni, variabili, comandi, controllo dell'esecuzione

La modularità degli algoritmi suggerisce anche un modo di intendere la programmazione, cioè a dettagli successivi. Ogni compito (o blocco) può essere implementato con una procedura (sub) o una funzione (function), cioè una parte di codice che esegue un unico compito (magari riutilizzabile in varie parti del programma).

```
Sub inizia()
    i=0
End sub
```

La funzione differisce da una procedura perché ritorna dei valori all'uscita della funzione.

In ogni momento il programma, oltre ad avere una posizione nel flusso, ha uno stato che dipende dai valori delle variabili. Le variabili sono come dei cassetti, dei luoghi nella memoria, che vengono utilizzati per modificarne i valori e poterli utilizzare. Ogni variabile è di un tipo (intero, stringa, numero reale, matrici di variabili,..) e viene dichiarata prima di essere utilizzata (per riservare spazio nella memoria).

```
Dim i as integer
Dim a1 as string
```

Mentre *i* è il nome della variabile, *integer* indica il tipo di variabile. E' possibile creare una variabile matrice (array, es. *dim ar(30) as integer*), così come è possibile creare strutture di tipi (type) che contengono più tipi:

```
type tipoPC
    dim ip as string
    dim sernum as integer
end type
dim primocomputer as tipoPC
```

Infine per implementare il flusso del programma si usano i comandi controllo di esecuzione come GO TO, FOR NEXT, DO WHILE, IF THEN ELSE. Tutti questi comandi realizzano il flusso logico del programma e sono comuni a tutti i linguaggi di programmazione. Per questo motivo l'importante è imparare a programmare con un linguaggio, per gli altri basta imparare solo le differenze per cui è molto più semplice.

L'ambiente di programmazione visuale: visual studio, documentazione in linea, compilazione, il progetto, i form, i moduli

Un tempo la programmazione modulare partiva dall'impostazione logica del programma nella quale l'interfaccia utente era solo una parte. Con lo svilupparsi delle interfacce grafiche e il massiccio riutilizzo di librerie preconfezionate, l'importanza della programmazione grafica (visuale) è diventata tale che a volte (specie in visual basic) si parte a costruire un programma dall'interfaccia grafica.

L'ambiente di programmazione: è il programma che consente di scrivere il codice, costruire il progetto (che è l'insieme dei sorgenti, cioè i file di partenza della compilazione), modificare e creare le form (le finestre del programma) e i controlli grafici, infine di compilare ed eseguire il programma. Visual studio comprende gli ambienti di programmazione microsoft visual basic, visual C++, java, visual foxpro ed ha eliminato il manuale fisico per sostituirlo con un sola documentazione in linea MSDN (developer network) sempre accessibile con la guida dei programmi (tasto F1 contestuale, cioè se si seleziona un comando e si lancia la guida viene visualizzata la guida su quel comando).

Ogni programma è rappresentato dal progetto, composto dai form (file .frm) e dai moduli che sono il codice generale (file .bas).

Nei form si progetta la parte grafica, si inseriscono i controlli (gli oggetti grafici come bottoni, campi di testo,…), e si scrive anche la parte di codice legata agli eventi degli oggetti grafici e dei form. Il codice (cioè le variabili generali, le procedure e le funzioni) che fanno parte di un form non sono però visibili (cioè non si possono utilizzare) da altri form o dal codice generale (contenuto nei moduli).

Il primo giro completo dell'ambiente di sviluppo è quindi: creare e salvare un progetto nuovo, con un form e un modulo (salvando anch'essi in file), aggiungere qualche controllo al form, scrivere del codice di un evento di un controllo (click di un bottone)

```
Sub Command1_click()
Msgbox "cliccato!"
Visualizza_i
End sub
```

Dichiarare le variabili globali nel modulo e scrivere una procedura globale e un'altra procedura che la chiama

```
Dim i as integer
Sub incrementa()
    i=i+1
End sub
Sub visualizza_i()
    'riga di commento: chiamata alla procedura incrementa
    incremata
    Msgbox "ora la variabile i e:"+str(i)
End sub
```

Infine compilare il programma ed eseguirlo sia in modalità debug che lanciando l'eseguibile creato.

Debugging di un programma: seguire un algoritmo passo passo

Quando si esegue il programma dall'ambiente di sviluppo (modalità debug, che significa eliminazione dei bachi) è possibile seguire passo passo il flusso di programmazione e controllare lo stato del programma (la visualizzazione della grafica e il valore delle variabili). Dato che è caricato in memoria l'ambiente di sviluppo, tutta la parte di interprete e molte librerie di debug sono già in esecuzione, per cui l'avvio in questa modalità è immediato mentre

la creazione di un eseguibile è molto più lunga. Per questo motivo il visual basic è molto comodo, anche all'interno delle applicazioni (VBA, visual basic for application) si può testare immediatamente la bontà del codice appena scritto. Inoltre quando si utilizza il programma in questa modalità è possibile interrompere l'esecuzione come in pausa e controllare lo stato del programma.

Programmazione ad oggetti: visual basic come esempio di programmazione ad oggetti grafici

Oltre alla preponderanza della parte grafica la programmazione di oggi è caratterizzata dagli oggetti (programmazione ad oggetti). Il java (le classi java), il C++ (i più stanno per aggiunta di oggetti) e lo stesso concetto di visual (cioè con aggiunta di oggetti grafici già pronti, i controlli) sono basati sul concetto di oggetto. Semplicisticamente un oggetto è come un tipo di cui non si definisce solo delle variabili (proprietà), ma anche delle procedure (metodi) e degli eventi a cui può rispondere.

Ad esempio estendendo il tipo precedente:

```
oggetto PC
proprietà
    ip as string
    sernum as integer
metodi
    connetti_al_server()
eventi
    su_accensione()
```

Per utilizzare un oggetto si crea un'istanza come si definisce una variabile di quel tipo (dim miopc as PC) e con il punto si ottengono sia le proprietà, sia i metodi (miopc.ip="192.168.0.0", miopc.connetti_al_server). I controlli sono degli oggetti grafici già a pronti e hanno tutte queste caratteristiche. Il bottone ha una serie di proprietà (accedibili in fase di progettazione con la finestra proprietà, in fase di esecuzione, cioè runtime, tramite il codice) come caption, enabled; ha metodi come move(); eventi come click().

```
Command1.caption="premi qui"
Command1.enabled=true
Command1.move 10,10
Sub command1_click()
'qui dentro i comandi quando viene premuto il pulsante
End sub
```

Utilizzo dei controlli: form, textbox, command button, primo esempio di programma

Un primo esempio di programmazione prevede una finestra principale (frmprova), tre text box per il calcolo della somma e un bottone per eseguirla. Nell'evento click del bottone va scritto il codice per eseguire la somma:

```
Private Sub cmdsomma_Click()
  txttot.Text = Val(txt_Iaddendo.Text) + Val(txt_IIaddendo.Text)
End Sub
```

cmdsomma è il nome di quel partiolare oggetto command button, ovvero cmdsomma è un'istanza di un oggetto bottone. Idem, txttot, txt_Iaddendo, txt_Iiaddendo sono 3 istanze dell'oggetto textbox. Si posso così vedere nella pratica le caratteristiche di questi semplici oggetti:

cmdsomma (oggetto command button): evento click

txttot (oggetto textbox): proprietà text

Il simbolo di uguale serve per assegnare una variabile: con a=b viene preso il valore di b e assegnato alla variabile a. Dato che le proprietà di un oggetto sono variabili, l'istruzione assegna alla prorpietà text dell'istanza txttot (tipo di dato: stringa) alla somma numerica delle proprietà text delle altre 2 istanze. Siccome la proprietà text è di tipo stringa, è necessario convertire il dato in numerico per poter fare la somma numerica, con la funzione val(). La funzione val() riceve come input (parametro) una stringa e dà come output (valore di ritorno) il valore numerico: ad esempio val("34a") è 34.

Esteso questo esempio ad una calcolatrice, all'evento click del bottone enter, si può associare una procedura simile utilizzando una serie di istruzioni if o più elegantemente utilizzando l'istruzione case:

```
Private Sub cmdrun_Click()
  num2 = Val(Text1.Text)
  If opr = "+" Then
    Text1.Text = num1 + num2
  Else
    If opr = "-" Then
      Text1.Text = num1 - num2
      Else
        If opr = "*" Then
            Text1.Text = num1 * num2
            Else
             If opr = "/" Then
                 Text1.Text = num1 / num2
             End If
        End If
    End If
  End If
End Sub
```

```
Private Sub cmdrun_Click()
  num2 = Val(Text1.Text)
  Select Case opr
  Case "+"
    Text1.Text = num1 + num2
  Case "-"
    Text1.Text = num1 - num2
  Case "*"
    Text1.Text = num1 * num2
  Case "/"
    Text1.Text = num1 / num2
  End Select
End Sub
```

Vengono utilizzate le variabili *num1* e *num2* per memorizzare i due numeri su cui eseguire le operazioni e la variabile *opr* per memorizzare l'istruzione precedentemente selezionata. Per essere visibili in tutte le procedure della form queste dichiarazioni devono essere inserite nella parte (generale) (dichiarazioni) del codice della form

```
Dim num1, num2 As Double
Dim opr As String
```

Manipolare i progetti

Un progetto è un insieme di file sorgenti, le risorse (come icone, file,..) e parametri di compilazione e di esecuzione. Manipolare un progetto significa quindi aggiungere o togliere file al progetto e modificare le proprietà del progetto.

Form: dialogo, SDI, MDI

I form possono essere di vario tipo:

Dialogo: è una finestra di dialogo, quindi non permette di passare ad altre finestre del programma finchè non si sceglie una azione

Single Document Interface: si indica una tipologia di utilizzo delle finestre per cui si aprono singolarmente

Multiple Document Interface: la tipologia di utilizzo delle finestre per cui esiste la finestra principale (main) e le finestre documento (child)) si aprono all'interno di questa.

Utilizzo dei controlli: accedere alle proprietà, metodi, eventi

Per aggiungere un controllo bisogna aggiungere il componente relativo (activex), si troverà quindi disponibile nella toolbox. I controlli possono essere anche creati e modificati a partire dai sorgenti che ne definiscono le proprietà, i metodi, gli eventi e l'apparenza grafica.
Per ogni istanza di un controllo (ovvero un elemento particolare dell'oggetto) si possono definire o modificare le proprietà (finestra proprietà) e si può scrivere il codice relativo ad un evento dell'oggetto, come ad esempio il *controllo.click()*, oppure definire nuovi metodi associati all'oggetto.

Menu, convenzioni di nomina di oggetti

I menu vengono creati con il menu editor e possono essere associati a qualsiasi finestra. All'interno del menu vengono anche definite le procedure relative ad ogni voce del menu.
Convenzioni per la nomina degli oggetti:

form	*frm*
check box	*chk*
combo box	*cbo*
command button	*cmd*
directory list box	*dir*
drive list box	*drv*
file list box	*fil*
frame	*fra*
grid	*grd*
horizontal scroll bar	*hsb*
image	*img*
label	*lbl*
line	*lin*
list box	*lst*
menu	*mnu*
option button	*opt*
OLE client	*ole*
Picture box	*pic*
Shape	*shp*
Text box	*txt*
Timer	*tmr*
Vertical scroll bar	*vsb*

Oggetti e istanza di un oggetto, collezioni e oggetti di sistema

L'instanza di un oggetto è la realizzazione particolare di un oggetto. L'istanza quindi di un oggetto tipo textbox è il particolare oggetto textbox chiamato *text1*.
Si possono creare istanze di oggetti anche con il codice e quindi runtime (durante il funzionamento del programma), e possono essere anche definiti come singoli elementi di un array di oggetti . Ad esempio in un progetto MDI, si possono creare finestre con un array di form child.

Variabili e costanti: scope, datatype, array, structure

Lo scope di una variabile indica la visibilità, ovvero all'interno di quale procedura viene vista la variabile e quindi dove è utilizzabile.
Visibilità di variabili

Dichiarazione interna alla procedura	la variabile è visibile solo dentro la form
Dichiarazione in (generale) (dichiarazioni) di una form	la variabile è visibile in tutte le procedure della form
Dichiarazione in un modulo con dim	la variabile è visibile in tutte le procedure del modulo
Dichiarazione in un modulo con global	la variabile è visibile in tutte le procedure del progetto

Il datatype indica il tipo di variabile.

Tipi di variabile, funzione vartype()

0 Empty (uninitialized)
1 Null (no valid data)
2 **Integer** : intero -32,768 to 32,767
3 Long integer : intero 4-byte -2,147,483,648 to 2,147,483,647
4 Single-precision floating-point number
5 **Double**-precision floating-point number : numero reale 64-bit
6 Currency value
7 Date value
8 **String** : sequenza di caratteri
9 Object
10 Error value
11 Boolean value
12 Variant (used only with arrays of variants)
13 A data access object
14 Decimal value
17 Byte value
36 Variants that contain user-defined types
8192 Array

L'array è una variabile matrice ad una o più dimensioni. L'accesso ad un elemento dell'array avviene come per le matrici arr[1,2] : colonna 1 riga 2

Procedure: Sub, function, chiamate a funzioni. Procedure in moduli o in form

Il codice può essere scritto o all'interno di un form oppure in un modulo. Le procedure in un modulo sono visibili nel codice di tutti i form e di tutti i moduli, mentre all'interno del form le procedure sono visibili solo in quel form. I codici relativi ai controlli si trovano nel form che li contiene.

Operatori, comandi di controllo esecuzione

Gli operatori sono funzioni predefinite che si applicano ad un oggetto o variabile, ad esempio operatori aritmetici come +, logici come and, di comparazione come <=.

I comandi di controllo esecuzione sono di decisione:

If...Then
If...Then...Else
Select Case

O di loop:

Do...Loop
For...Next
For Each...Next

Debug e gestione degli errori

Per debug si intende la correzioni degli errori (togliere i bachi) e gli strumenti per analizzare lo stato del programma in esecuzione (run-time). Nel codice si possono inserire punti di interruzione (breakpoint) in modo che il programma si fermi in un determinato punto per poter analizzare ogni variabile o istruzione. A quel punto è possibile eseguire una istruzione alla volta per controllare la variazione dello stato del programma (grafica, variabili, ..) ad ogni istruzione, oppure continuare l'esecuzione fino al prossimo punto di interruzione.

Connessione a database access. Utilizzo del recordset

Un recordset è un insieme di record (righe) risultanti da una richiesta ad un database. Il dynaset è un recordset dinamico per cui è possibile anche modificare i record ottenuti.

```
Dim r1 As dynaset
Set r1=db.createDynaset("select indirizzo from anag where tel like'*02*')
R1.movefirst
R1("indirizzo")="milano"
```

Creazione di installazioni, componenti aggiuntive (activex)

Per creare le installazioni che installano sia il programma creato (.exe), sia le .dll e componenti (activex) necessari al programma, si utilizza la creazione guidata di installazioni a partire dal progetto. Vengono riconosciuti i componenti necessari e inclusi nell'installazione, creato il file setup.exe e compressi tutti i file di installazione.
I componenti aggiuntivi sono activex già installati o installabili che consentono di utilizzare controlli più complessi o particolari rispetto a quelli predefiniti.

Caratteri ANSI

0	•	32	[space]	64	@	96	`
1	•	33	!	65	A	97	a
2	•	34	"	66	B	98	b
3	•	35	#	67	C	99	c
4	•	36	$	68	D	100	d
5	•	37	%	69	E	101	e
6	•	38	&	70	F	102	f
7	•	39	'	71	G	103	g
8	* *	40	(	72	H	104	h
9	* *	41	)	73	I	105	i
10	* *	42	*	74	J	106	j
11	•	43	+	75	K	107	k
12	•	44	,	76	L	108	l
13	* *	45	-	77	M	109	m
14	•	46	.	78	N	110	n
15	•	47	/	79	O	111	o
16	•	48	0	80	P	112	p
17	•	49	1	81	Q	113	q
18	•	50	2	82	R	114	r
19	•	51	3	83	S	115	s
20	•	52	4	84	T	116	t
21	•	53	5	85	U	117	u
22	•	54	6	86	V	118	v

23	•	55	7	87	W	119	w
24	•	56	8	88	X	120	x
25	•	57	9	89	Y	121	y
26	•	58	:	90	Z	122	z
27	•	59	;	91	[	123	{
28	•	60	<	92	\	124	|
29	•	61	=	93	]	125	}
30	•	62	>	94	^	126	~
31	•	63	?	95	_	127	•

128	€	160	[space]	192	À	224	à
129	•	161	¡	193	Á	225	á
130	‚	162	¢	194	Â	226	â
131	*ƒ*	163	£	195	Ã	227	ã
132	„	164	¤	196	Ä	228	ä
133	…	165	¥	197	Å	229	å
134	†	166	¦	198	Æ	230	æ
135	‡	167	§	199	Ç	231	ç
136	ˆ	168	¨	200	È	232	è
137	‰	169	©	201	É	233	é
138	Š	170	ª	202	Ê	234	ê
139	‹	171	«	203	Ë	235	ë
140	Œ	172	¬	204	Ì	236	ì
141	•	173		205	Í	237	í
142	Ž	174	®	206	Î	238	î

143	•	175	¯	207	Ï	239	ï
144	•	176	°	208	Ð	240	ð
145	‘	177	±	209	Ñ	241	ñ
146	’	178	²	210	Ò	242	ò
147	“	179	³	211	Ó	243	ó
148	”	180	´	212	Ô	244	ô
149	•	181	µ	213	Õ	245	õ
150	–	182	¶	214	Ö	246	ö
151	—	183	·	215	×	247	÷
152	˜	184	¸	216	Ø	248	ø
153	™	185	¹	217	Ù	249	ù
154	š	186	º	218	Ú	250	ú
155	›	187	»	219	Û	251	û
156	œ	188	¼	220	Ü	252	ü
157	•	189	½	221	Ý	253	ý
158	ž	190	¾	222	Þ	254	þ
159	Ÿ	191	¿	223	ß	255	ÿ

• These characters aren't supported by Microsoft Windows.

* *Values 8, 9, 10, and 13 convert to backspace, tab, linefeed, and carriage return characters, respectively. They have no graphical representation but, depending on the application, can affect the visual display of text. The values in the table are the Windows default. However, values in the ANSI character set above 127 are determined by the code page specific to your operating system.