

INFORMATICA



MODULO 8

Algoritmi e programmi

A cura di Mimmo Corrado

Aprile 2011

FINALITÀ

Il Modulo Algoritmi e programmi introduce lo studente nel mondo della programmazione.



1. FLOW-CHART E PSEUDOLINGUAGGIO

1.1 Definire il termine "algoritmo"

Il termine "algoritmo" deriva dal nome del matematico arabo *Al-Khwarizmi*, vissuto nell'800 d.C., ritenuto l'ideatore del procedimento che consente di effettuare il calcolo della moltiplicazione tra due numeri mediante l'incolonnamento delle cifre (quella che ancora oggi usiamo).

Nel senso più ampio della parola, un "algoritmo" è una sequenza finita di operazioni, come ad esempio una ricetta di cucina, o le istruzioni di funzionamento di una lavatrice.

In informatica, con il termine algoritmo si intende: **un procedimento (sequenza finita di istruzioni elementari) per la risoluzione di un problema, rappresentato in un linguaggio comprensibile all'uomo e adatto ad essere tradotto in un programma eseguibile da un computer.**



Al-Khwarizmi

Un algoritmo deve:

- ✚ essere **finito**: la sequenza di istruzioni deve essere finita e portare ad un risultato avere un punto di *Inizio*, dove si avvia l'esecuzione delle azioni, e un punto di *Fine*, dove si interrompe l'esecuzione
- ✚ essere **eseguibile**: le istruzioni devono poter essere eseguite materialmente dall'esecutore
- ✚ essere **non ambiguo**: le istruzioni devono essere espresse in modo tale da essere interpretate da tutti allo stesso modo
- ✚ essere **generale**: deve essere valido non solo per un particolare problema, ma per una classe di problemi
- ✚ essere **deterministico**: partendo dagli stessi dati iniziali deve portare sempre allo stesso risultato finale indipendentemente dall'esecutore
- ✚ essere **completo**: deve contemplare tutti i casi che si possono verificare durante l'esecuzione

1.2 Descrivere in forma algoritmica la procedura risolutiva di semplici problemi

Come accennato precedentemente, gli algoritmi li incontriamo e li eseguiamo quotidianamente:

- ✚ nella preparazione di un uovo al tegamino
- ✚ nella messa in moto dell'auto
- ✚ nel calcolo dell'area del rettangolo
- ✚ nel calcolo dell'M.C.D. fra due numeri naturali
- ✚ nel trasporto del lupo, della pecora e del cavolo da una sponda all'altra di un fiume
- ✚ ecc...

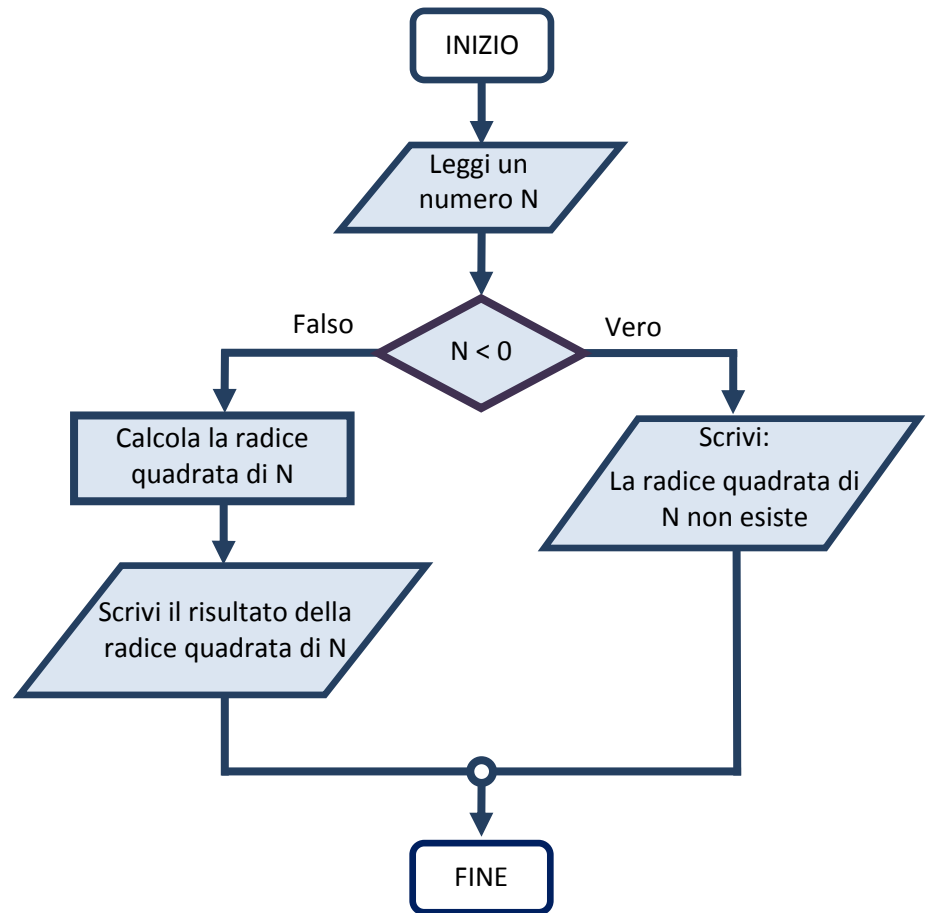
Ad esempio l'algoritmo risolutivo dell'ultimo problema, ricordando che il lupo non mangia il cavolo, è il seguente:

1. Inizio
2. traghettare la pecora sulla sponda B, lasciando assieme il lupo ed il cavolo
3. ritornare con la barca vuota sulla sponda A, lasciando la pecora da sola sulla sponda B
4. traghettare il cavolo sulla sponda B, lasciando il lupo da solo sulla sponda A
5. riportare la pecora sulla sponda A, lasciando il cavolo da solo sulla sponda B
6. traghettare il lupo sulla sponda B, lasciando da sola la pecora sulla sponda A
7. ritornare con la barca vuota sulla sponda A
8. traghettare la pecora sulla sponda B
9. Fine

1.3 Rappresentare algoritmi mediante diagrammi

Un modo chiaro per descrivere un algoritmo è rappresentato dai **diagrammi di flusso** (flow chart o diagrammi a blocchi).

I diagrammi di flusso sono grafici creati utilizzando una successione di figure (ognuna delle quali identifica una particolare azione) che rappresentano un ragionamento logico di immediata comprensione.



I simboli utilizzati in un diagramma di flusso sono i seguenti:

Oggetto grafico	Denominazione	Significato
	Punto di partenza	Rappresenta un'azione che avvia il processo
	Punto di fine	Rappresenta un'azione che conclude il processo
	Leggi	Rappresenta un'azione di ingresso dati
	Scrivi	Rappresenta un'azione di uscita dei risultati
	Elaborazione	Rappresenta il comando o calcolo da eseguire
	Test	Rappresenta la scelta fra due possibili percorsi
	Linea di flusso	Indica la direzione del percorso del flusso
	Connessione	Rappresenta il punto d'inserimento nel grafico (generalmente contiene una lettera o un numero d'ordine)

1.4 Scrivere un semplice programma con l'uso del pseudolinguaggio

Un altro modo per rappresentare gli algoritmi consiste nell'uso di uno pseudolinguaggio. Questo tipo di linguaggio descrive le istruzioni con frasi rigorose anziché con i simboli grafici: si utilizzano parole chiave, operatori e nomi di variabili.

Tuttavia, questo tipo di linguaggio non è direttamente comprensibile dai programmi compilatori e interpreti e dovrà quindi essere tradotto in linguaggio di alto livello.

Non esiste un unico pseudo linguaggio. Analisti e programmatori di una azienda utilizzano uno pseudo linguaggio diverso da quello usato dai colleghi di un'altra.

Esempio

Pseudolinguaggio che traduce l'algoritmo per il calcolo dell'area di un rettangolo

PROGRAMMA Area del Rettangolo

INIZIO

LEGGI (base)

LEGGI (altezza)

AREA = base · altezza

SCRIVI (area)

FINE

2.0 PROGRAMMAZIONE STRUTTURATA

2.1 La programmazione strutturata

La **programmazione strutturata** è un paradigma (metodologia) di programmazione emerso nella seconda metà degli anni '60, che ha introdotto i concetti fondamentali che sono alla base di tutti gli altri paradigmi successivi, compresi gli ultimi linguaggi orientati agli oggetti.

Un paradigma di programmazione è uno stile fondamentale di programmazione, ovvero un insieme di strumenti concettuali forniti da un linguaggio di programmazione per la stesura dei programmi.

Esso è nato dalla necessità di portare ordine e criteri di lavoro più efficienti nella produzione dei programmi. Occorreva trovare una metodologia alternativa alla programmazione basata sul **salto incondizionato** (o goto) dei primi linguaggi di programmazione, che rendeva il codice scritto praticamente incomprensibile dallo stesso autore, tanto da essere chiamato **spaghetti code**, per la sua natura ingarbugliata.

La programmazione strutturata è una tecnica di programmazione che limita l'utilizzo delle sole tre strutture fondamentali:

- ✚ Struttura di sequenza
- ✚ Struttura di selezione (o alternativa)
- ✚ Struttura di iterazione (o ciclo, ripetizione).

Con questa tipologia di programmazione, la descrizione degli algoritmi viene effettuata in modo chiaro, facilmente leggibile e comprensibile: tale codice può essere agevolmente compreso e modificato in un secondo tempo anche da un eventuale programmatore che non sia il suo creatore.

I linguaggi di programmazione strutturati iniziarono a emergere nei primi anni 70, facendo tesoro delle idee dei due matematici italiani Corrado Bohm e Giuseppe Jacopini che per primi sottolinearono, in un teorema, come le tre strutture fondamentali offrivano un insieme di strutture di controllo completo, che garantivano la possibilità di descrivere tutti gli algoritmi.

Teorema di Jacopini-Bohm (1966)

Un qualsiasi algoritmo può essere espresso utilizzando esclusivamente le tre strutture di controllo: sequenza, selezione e iterazione.

Fra i linguaggi tipici del paradigma strutturato si possono citare:

- ✚ il linguaggio **Pascal**, elaborato nel 1968 dal professor Wirth del Politecnico di Zurigo, che è tutt'oggi il linguaggio più diffuso e utilizzato per affrontare lo studio della programmazione
- ✚ il linguaggio **C**, messo a punto da Dennis Ritchie per implementare i primi sistemi operativi negli anni '70, che è il linguaggio di riferimento sia per i programmatori "più esperti" sia per i moderni linguaggi di programmazione dell'ambiente Web (**C++**, **Java**, **PHP** ecc. usano la stessa sintassi propria del linguaggio C).
- ✚ il Cobol
- ✚ l'algol

2.2 La struttura di sequenza

La struttura di sequenza è l'elenco ordinato delle istruzioni da eseguire.

In un ogni algoritmo, in quanto tale, non manca mai la struttura di sequenza.

Struttura di sequenza	
Flow-chart	Pseudolinguaggio
<pre> graph TD INIZIO([INIZIO]) --> I1[/Istruzione 1/] I1 --> I2[/Istruzione 2/] I2 --> I3[Istruzione 3] I3 --> I4[/Istruzione 4/] I4 --> FINE([FINE]) </pre>	INIZIO Istruzione 1 Istruzione 2 Istruzione 3 Istruzione 4 FINE

Esempio – Struttura di sequenza

Dati la misura della base b e dell'altezza h di un rettangolo, calcola l'area del rettangolo S .

Area del rettangolo						
Flow-chart	Pseudolinguaggio		Trace table (Input: b=3 ; h=2)			
	n°	Istruzione	n°	b	h	S
<pre> graph TD INIZIO([INIZIO]) --> L1[/Leggi b/] L1 --> L2[/Leggi h/] L2 --> A["S = b · h"] A --> S[/Scrivi S/] S --> FINE([FINE]) </pre>	1	INIZIO	1			
	2	Leggi la base b	2	3		
	3	Leggi l'altezza h	3		4	
	4	Assegna a S il valore $b \cdot h$	4			$3 \cdot 2 = 6$
	5	Scrivi S	5			6
	6	FINE	6			

2.3 La struttura di selezione

In un algoritmo si può verificare che le istruzioni da eseguire siano diverse a seconda dei dati elaborati.

L'esecutore deve effettuare una scelta fra due alternative. Per effettuare tale scelta l'esecutore controlla il grado di verità della **condizione** posta in un test.

Se la condizione risulta **Vera** l'esecutore esegue l'istruzione A.

Se la condizione risulta **Falsa** l'esecutore segue l'istruzione B.

Struttura di selezione	
Flow-chart	Pseudolinguaggio
<pre> graph TD Start(()) --> Cond{Condizione} Cond -- F --> IB[Istruzione B] Cond -- V --> IA[Istruzione A] </pre>	<p>Se Condizione allora Istruzione A altrimenti Istruzione B</p>

Codifica in linguaggio Visual Basic	
Selezione a due uscite	Selezione a una uscita
<p>IF Condizione = Vera THEN Istruzione A ELSE Istruzione B END IF</p>	<p>IF Condizione = Vera THEN Istruzione A END IF</p>

Esempio – Struttura di selezione

Dati due numeri **a** e **b**, calcola il maggiore.

Max fra due numeri																																																				
Flow-chart		Pseudolinguaggio		Trace table (Input: a=3 ; b=5)																																																
<pre> graph TD Start((INIZIO)) --> ReadA[/Leggi a/] ReadA --> ReadB[/Leggi b/] ReadB --> Cond{a > b} Cond -- F --> WriteB[Scrivi b] Cond -- V --> WriteA[Scrivi a] WriteB --> End((FINE)) WriteA --> End </pre>		<table border="1"> <thead> <tr> <th>n°</th> <th>Istruzione</th> <th>n°</th> <th>a > b</th> <th>a</th> <th>b</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>INIZIO</td> <td>1</td> <td></td> <td colspan="2">INIZIO</td> </tr> <tr> <td>2</td> <td>Leggi il numero <i>a</i></td> <td>2</td> <td></td> <td>3</td> <td></td> </tr> <tr> <td>3</td> <td>Leggi il numero <i>b</i></td> <td>3</td> <td></td> <td></td> <td>5</td> </tr> <tr> <td>4</td> <td>Se $a > b$</td> <td>4</td> <td>Falso</td> <td></td> <td></td> </tr> <tr> <td>5</td> <td> allora Scrivi <i>a</i></td> <td>5</td> <td></td> <td></td> <td></td> </tr> <tr> <td>6</td> <td> altrimenti Scrivi <i>b</i></td> <td>6</td> <td></td> <td></td> <td>5</td> </tr> <tr> <td>7</td> <td>FINE</td> <td>7</td> <td></td> <td colspan="2">FINE</td> </tr> </tbody> </table>	n°	Istruzione	n°	a > b	a	b	1	INIZIO	1		INIZIO		2	Leggi il numero <i>a</i>	2		3		3	Leggi il numero <i>b</i>	3			5	4	Se $a > b$	4	Falso			5	allora Scrivi <i>a</i>	5				6	altrimenti Scrivi <i>b</i>	6			5	7	FINE	7		FINE			
n°	Istruzione	n°	a > b	a	b																																															
1	INIZIO	1		INIZIO																																																
2	Leggi il numero <i>a</i>	2		3																																																
3	Leggi il numero <i>b</i>	3			5																																															
4	Se $a > b$	4	Falso																																																	
5	allora Scrivi <i>a</i>	5																																																		
6	altrimenti Scrivi <i>b</i>	6			5																																															
7	FINE	7		FINE																																																

2.4 La struttura di iterazione

In un algoritmo può capitare che alcune istruzioni debbano essere ripetute, in modo identico, più volte.

La ripetizione di un insieme di istruzioni prende il nome di *iterazione* o *ciclo* (loop).

Il gruppo di istruzioni ripetute prende il nome di *corpo del ciclo*.

L'istruzione di iterazione può essere definita o indefinita.

L'iterazione è detta *definita* quando è noto a priori il numero di ripetizioni.

L'iterazione è detta *indefinita* quando il ciclo viene ripetuto un numero di volte sconosciuto a priori e termina quando si verifica una determinata condizione.

Esempi

- 🚧 Ripeti 10 volte la preghiera "Atto di dolore" (Iterazione definita)
- 🚧 Bevi un bicchiere di vino finché non cadi a terra ubriaco (Iterazione indefinita)

L'iterazione indefinita è detta *precondizionale* (o iterazione per vero) se il controllo per l'arresto dell'iterazione è posto prima del gruppo di istruzioni da ripetere.

L'iterazione indefinita è detta *postcondizionale* (o iterazione per falso) se il controllo per l'arresto dell'iterazione è posto dopo del gruppo di istruzioni da ripetere.

Strutture di Iterazione

<i>Iterazione indefinita precondizionale</i>	<i>Iterazione indefinita postcondizionale</i>	<i>Iterazione definita enumerativa</i>
Mentre <i>Condizione = Vera</i> esegui Istruzioni	Ripeti Istruzioni finché <i>Condizione = Falsa</i>	Ripeti Istruzioni N volte
Esce dal ciclo quando <i>Condizione = Falsa</i>	Esce dal ciclo quando <i>Condizione = Vera</i>	Esce dal ciclo quando <i>Contatore = N</i>

Codifica in linguaggio Visual Basic

<i>Iterazione indefinita precondizionale</i>	<i>Iterazione indefinita postcondizionale</i>	<i>Iterazione definita enumerativa</i>
Do While Condizione = Vera Istruzioni Loop	Do Istruzioni Loop Until Condizione = Falsa	For contatore = Iniziale To Finale Istruzioni Next

Esempio 1 - *Iterazione indefinita precondizionale*

Calcola la **Somma** dei primi **N** numeri naturali, senza utilizzare la formula di Gauss $S_n = \frac{n(n+1)}{2}$.

Somma dei primi N numeri naturali						
Flow-chart	Pseudolinguaggio	Trace table (Input N=5)				
	n°	Istruzione	n°	Cont < N	Contatore	Somma
<pre> graph TD INIZIO([INIZIO]) --> LeggiN[/Leggi N/] LeggiN --> Contatore0[Contatore = 0] Contatore0 --> Somma0[Somma = 0] Somma0 --> Decision{Contatore < N} Decision -- V --> ContatorePlus[Contatore = Contatore + 1] ContatorePlus --> SommaPlus[Somma = Somma + Contatore] SommaPlus --> Decision Decision -- F --> ScriviSomma[/Scrivi Somma/] ScriviSomma --> FINE([FINE]) </pre>	1	INIZIO	3		0	
	2	Leggi il numero <i>N</i>	4			0
	3	Assegna al <i>Contatore</i> il valore <i>0</i>	5	Vero		
	4	Assegna alla <i>Somma</i> il valore <i>0</i>	7		1	
	5	<i>Mentre</i> il <i>Contatore</i> < <i>N</i> <i>fai</i>	8			0+1=1
	6	<i>Inizio</i>	5	Vero		
	7	Incrementa di <i>1</i> il <i>contatore</i>	7		2	
	8	Aggiungi alla <i>somma</i> il <i>contatore</i>	8			1+2=3
	9	<i>Fine</i>	5	Vero		
	10	Scrivi <i>Somma</i>	7		3	
	11	FINE	8			3+3=6
			5	Vero		
			7		4	
			8			6+4=10
			5	Vero		
			7		5	
			8			10+5=15
			5	Falso		
			10			15
			11	FINE		

Esempio 2 - *Iterazione indefinita postcondizionale*

Calcola la **Somma** dei primi **N** numeri naturali, senza utilizzare la formula di Gauss $S_n = \frac{n(n+1)}{2}$.

Somma dei primi N numeri naturali						
Flow-chart	Pseudolinguaggio	Trace table (Input N=5)				
	n°	Istruzione	n°	Cont < N	Contatore	Somma
<pre> graph TD INIZIO([INIZIO]) --> LeggiN[/Leggi N/] LeggiN --> Contatore0[Contatore = 0] Contatore0 --> Somma0[Somma = 0] Somma0 --> LoopStart(()) LoopStart --> ContatoreInc[Contatore = Contatore + 1] ContatoreInc --> SommaAdd[Somma = Somma + Contatore] SommaAdd --> Decision{Contatore >= N} Decision -- V --> ScriviSomma[/Scrivi Somma/] ScriviSomma --> FINE([FINE]) Decision -- F --> LoopStart </pre>	1	INIZIO	1	INIZIO		
	2	Leggi il numero <i>N</i>	2			
	3	Assegna al <i>Contatore</i> il valore <i>0</i>	3		0	
	4	Assegna alla <i>Somma</i> il valore <i>0</i>	4			0
	5	<i>Ripeti</i>	7		1	
	6	<i>Inizio</i>	8			0+1=1
	7	Incrementa di <i>1</i> il <i>contatore</i>	10	Vero		
	8	Aggiungi alla <i>somma</i> il <i>contatore</i>	7		2	
	9	<i>Fine</i>	8			1+2=3
	10	<i>Finchè</i> <i>Contatore < N</i>	10	Vero		
	11	Scrivi <i>Somma</i>	7		3	
	12	FINE	8			3+3=6
			10	Vero		
			7		4	
			8			6+4=10
			10	Vero		
			7		5	
			8			10+5=15
			10	Falso		
			11			15
			12		FINE	

Esempio 3 - *Iterazione definita enumerativa*

Calcola la **Somma** dei primi **N** numeri naturali, senza utilizzare la formula di Gauss $S_n = \frac{n(n+1)}{2}$.

Somma dei primi N numeri naturali					
Flow-chart	Pseudolinguaggio	Trace table (N=5)			
<pre> graph TD INIZIO([INIZIO]) --> LeggiN[/Leggi N/] LeggiN --> Somma0[Somma = 0] Somma0 --> Contatore[Contatore = 1, N] Contatore --> SommaAdd[Somma = Somma + Contatore] SommaAdd --> Decision{ } Decision --> Contatore Decision --> ScriviSomma[/Scrivi Somma/] ScriviSomma --> FINE([FINE]) </pre>	1	INIZIO	3		0
	2	Leggi il numero <i>N</i>	4	1	
	3	Assegna alla <i>Somma</i> il valore <i>0</i>	6		0+1=1
	4	Ripeti con <i>Contatore da 1 a N</i>	4	2	
	5	<i>Inizio</i>	6		1+2=3
	6	Aggiungi alla <i>somma</i> il <i>contatore</i>	4	3	
	7	<i>Fine</i>	6		3+3=6
	8		4	4	
	9	Scrivi <i>Somma</i>	6		6+4=10
	10	FINE	4	5	
		6		10+5=15	
		9		15	
		10		FINE	

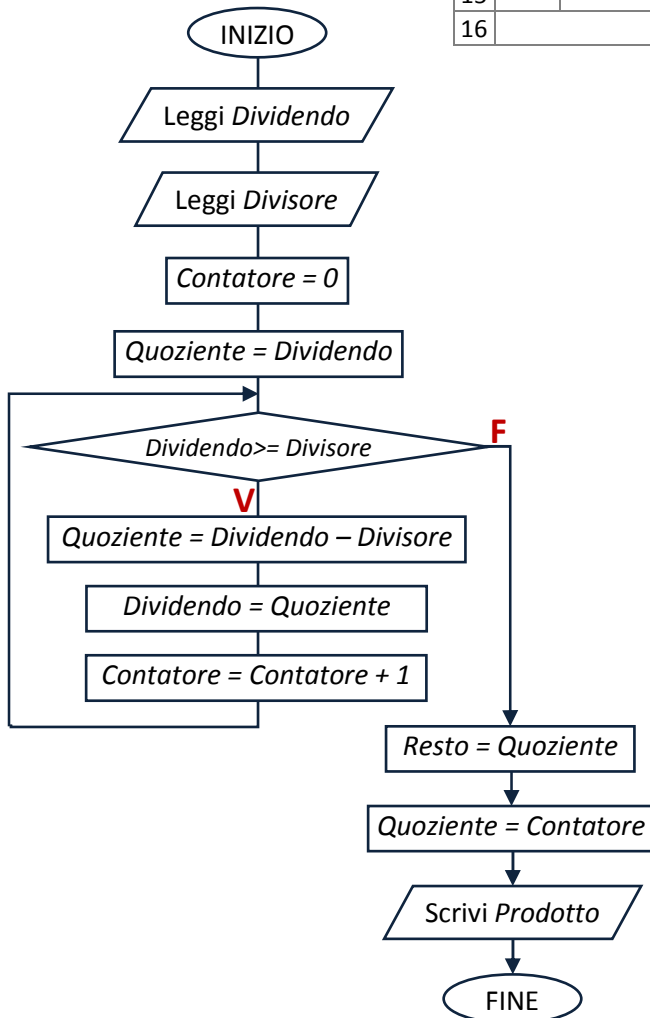
Esempio 4 - *Iterazione indefinita precondizionale*

Dati due numeri *A* e *B*, calcola il **Prodotto** dei i due numeri utilizzando solo l'operazione di addizione.

Prodotto di due numeri					
Flow-chart	Pseudolinguaggio	Trace table (Input: A=6 ; B=4)			
	n° Istruzione	n°	Cont < B	Contatore Prodotto	
<pre> graph TD Start([INIZIO]) --> ReadA[/Leggi A/] ReadA --> ReadB[/Leggi B/] ReadB --> SetCont[Contatore = 0] SetCont --> SetProd[Prodotto = 0] SetProd --> Decide{Contatore < B} Decide -- V --> IncCont[Contatore = Contatore + 1] IncCont --> AddProd[Prodotto = Prodotto + A] AddProd --> Decide Decide -- F --> WriteProd[/Scrivi Prodotto/] WriteProd --> End([FINE]) </pre>	1 INIZIO	4		0	
	2 Leggi il numero <i>A</i>	5			0
	3 Leggi il numero <i>B</i>	6	Vero		
	4 Assegna al <i>Contatore</i> il valore <i>0</i>	8		1	
	5 Assegna al <i>Prodotto</i> il valore <i>0</i>	9			0+6=6
	6 <i>Mentre</i> il <i>Contatore</i> < <i>B</i> fai	6	Vero		
	7 <i>Inizio</i>	8		2	
	8 Incrementa di <i>1</i> il <i>contatore</i>	9			6+6=12
	9 Incrementa di <i>A</i> il <i>prodotto</i>	6	Vero		
	10 <i>Fine</i>	8		3	
	11 Scrivi <i>Prodotto</i>	9			12+6=18
	12 FINE	6	Vero		
		8		4	
		9		18+6=24	
		6	Falso		
		11		24	
		12	FINE		

Esempio 5 - *Iterazione indefinita precondizionale*

Divisione con resto fra due numeri								
Pseudolinguaggio		Trace table (Input: Dividendo =19 ; Divisore=5)						
n°	Istruzione	n°	Test	Contatore	Dividendo	Divisore	Quoziente	Resto
1	INIZIO	1						
2	Leggi il <i>Dividendo</i>	2			19			
3	Leggi il <i>Divisore</i>	3				5		
4	Assegna al <i>Contatore</i> il valore 0	4		0				
5	Assegna al <i>Quoziente</i> il valore <i>Dividendo</i>	5					19	
6	<i>Mentre</i> il <i>Dividendo</i> >= <i>Divisore</i> <i>fai</i>	6	Vero					
7	<i>Inizio</i>	8					19 - 5 = 14	
8	<i>Quoziente</i> = <i>Dividendo</i> - <i>Divisore</i>	9			14			
9	<i>Dividendo</i> = <i>Quoziente</i>	10		1				
10	Incrementa di 1 il <i>Contatore</i>	6	Vero					
11	<i>Fine</i>	8					14 - 5 = 9	
12	<i>Resto</i> = <i>Quoziente</i>	9			9			
13	<i>Quoziente</i> = <i>Contatore</i>	10		2				
14	Scrivi <i>Quoziente</i>	6	Vero					
15	Scrivi <i>Resto</i>	8					9 - 5 = 4	
16	FINE	9			4			
		10		3				
		6	Falso					
		12						4
		13					3	
		14					3	
		15						4
		16	Fine					



M.C.D. fra due numeri (Algoritmo di Euclide)

Per determinare il M.C.D. fra due numeri naturali m e n si può sfruttare il seguente teorema.

TEOREMA

Supposto $m \geq n$, se m e n hanno un divisore d comune, d è divisore anche di $m - n$.

Dimostrazione

Poiché d è divisore sia di m che di n si ha che $m = kd$ e $n = hd$

La differenza $m - n$ è allora: $m - n = kd - hd$

Raccogliendo il fattore d si ottiene $m - n = d(k - h)$.

Si deduce che anche $m - n$ ha d come divisore.

Pertanto i divisori comuni a m e n sono comuni anche a $m - n$ e n .

Cioè: $M.C.D.(m, n) = M.C.D.(m - n, n)$.

Si può allora determinare il M.C.D. fra due numeri per sottrazioni successive.

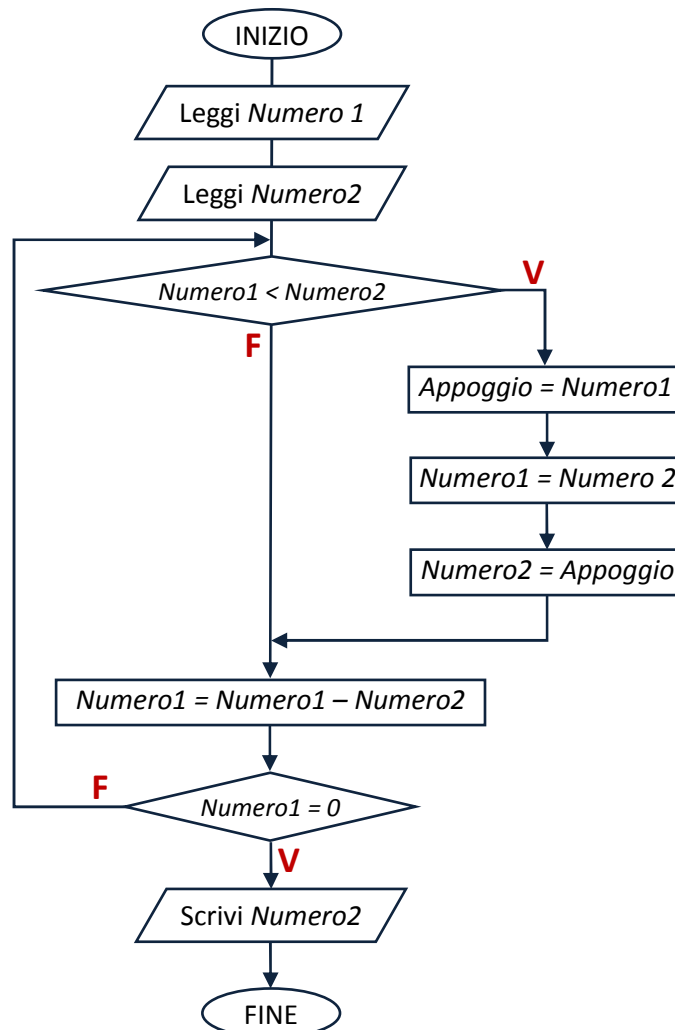
L'algoritmo è il seguente:

- ✚ si confrontano i due numeri, se il primo è minore del secondo si scambiano
- ✚ si esegue la sottrazione fra i due numeri
- ✚ si confronta poi il secondo numero con la differenza, se è necessario si scambiano
- ✚ si esegue la sottrazione fra i due numeri
- ✚ Si prosegue in questo modo fino ad ottenere una sequenza di numeri (*sempre più piccoli*) che hanno il medesimo M.C.D.
- ✚ Così facendo si giunge a 0, e a questo punto, essendo $M.C.D.(0, n) = n$, si conclude che il numero precedente è il M.C.D. cercato.

Nota

Per effettuare lo scambio dei numeri m e n occorre una terza variabile.

M.C.D. di Euclide (Sottrazioni successive)								
Pseudolinguaggio		Trace table (Input: Dividendo =6 ; Divisore=15)						
n°	Istruzione	n°	Test Se	Test Finchè	Numero1	Numero2	Appoggio	M.C.D.
1	INIZIO	1						
2	Leggi <i>Numero1</i>	2			6			
3	Leggi <i>Numero2</i>	3				15		
4	<i>Ripeti</i>	5	Vero					
5	<i>Se</i> Numero1 < Numero2 <i>Allora</i>	7					6	
6	<i>Inizio</i>	8			15			
7	Appoggio = Numero1	9				6		
8	Numero1 = Numero2	11			15 - 6 = 9			
9	Numero2 = Appoggio	12		Falso				
10	<i>Fine</i>	5	Falso					
11	Numero1 = Numero1 - Numero2	11			9 - 6 = 3			
12	<i>Finchè</i> Numero1 = 0	12		Falso				
13	<i>M.C.D.</i> = Numero2	5	Vero					
14	Scrivi <i>M.C.D.</i>	7					3	
15	FINE	8			6			
		9				3		
		11			6 - 3 = 3			
		12		Falso				
		5	Falso					
		11			3 - 3 = 0			
		12		Vero				
		13						3
		14						3
		15						Fine



3. VISUAL BASIC FOR APPLICATION

3.1 Linguaggi di programmazione

Un **linguaggio** è un insieme di parole (lessico o vocabolario) e della loro sintassi (regole da seguire per costruire una frase che risulti comprensibile dal ricevente).

Il linguaggio è un sistema di comunicazione esclusivo dell'uomo che gli permette di interagire con i suoi simili. Nelle diverse regioni del mondo si sono sviluppati, negli anni, linguaggi in differenti lingue definiti: linguaggi naturali.

Il **linguaggio naturale** dell'uomo consente ricchezza espressiva ma anche, alcune volte, **ambiguità**. Ad esempio la frase: "la giovane mente" può essere interpretata in differenti modi.

Il linguaggio naturale pertanto, non può essere utilizzato per "istruire" l'elaboratore su ciò che deve compiere, in quanto le caratteristiche che presenta non sono adatte alla sua logica.

Il computer riesce ad interpretare un linguaggio costituito da istruzioni non ambigue e ad esso comprensibili. L'unico linguaggio direttamente comprensibile dall'elaboratore è il linguaggio macchina.

Il **linguaggio macchina** ha una sintassi limitatissima. Esso è costituito da una serie di comandi in codice binario (sequenze di 0 e 1), programmati mediante la logica booleana, che impegnano direttamente i circuiti elettronici del computer. Inoltre è strettamente collegato alla struttura fisica del particolare elaboratore. Ciò vuol dire che un medesimo programma, scritto in linguaggio macchina, può non funzionare in microprocessori diversi.

Per ovviare a questi inconvenienti sono stati sviluppati i **linguaggi di programmazione** (di alto livello) in cui le istruzioni non sono più indicate da sequenze di cifre binarie, ma da nomi simbolici, più facili da riconoscere, memorizzare e utilizzare da parte del programmatore. Inoltre essi non sono vincolati alla struttura dello specifico processore, ma risultano funzionanti su qualsiasi elaboratore.

Tuttavia, poiché l'elaboratore riesce ad interpretare solo istruzioni formulate in linguaggio macchina, per ogni programma scritto in linguaggio di alto livello esiste il corrispondente programma traduttore in linguaggio macchina.

Questa operazione di traduzione può essere eseguita da:

- ✚ **Programmi compilatori**, che traducono l'intero programma scritto in linguaggio di alto livello (in Pascal, in C++,...) nella corrispondente copia in linguaggio macchina: tutte le istruzioni vengono controllate nel lessico e nella sintassi, tradotte e trasformate in un file eseguibile (con estensione .exe nei sistemi Windows) che, memorizzato per esempio in un CD o su disco fisso, potrà essere utilizzato tutte le volte che si ritiene opportuno. Il file eseguibile, quindi, diventa indipendente dal programma scritto in linguaggio di alto livello che lo ha prodotto e una volta tradotto può essere eseguito senza il programma compilatore.
- ✚ **Programmi interpreti**, che "leggono" riga per riga le istruzioni scritte in linguaggio di alto livello, ne controllano il lessico e la sintassi e le traducono in linguaggio macchina per farle eseguire direttamente dall'unità centrale di elaborazione. In questo caso non viene prodotta una copia del programma in linguaggio macchina, ma ogni istruzione viene di volta in volta tradotta e poi fatta eseguire.

Per rispondere alle diverse esigenze sono stati sviluppati diversi linguaggi di programmazione di alto livello (o procedurali), adatti ad ambiti applicativi diversi:

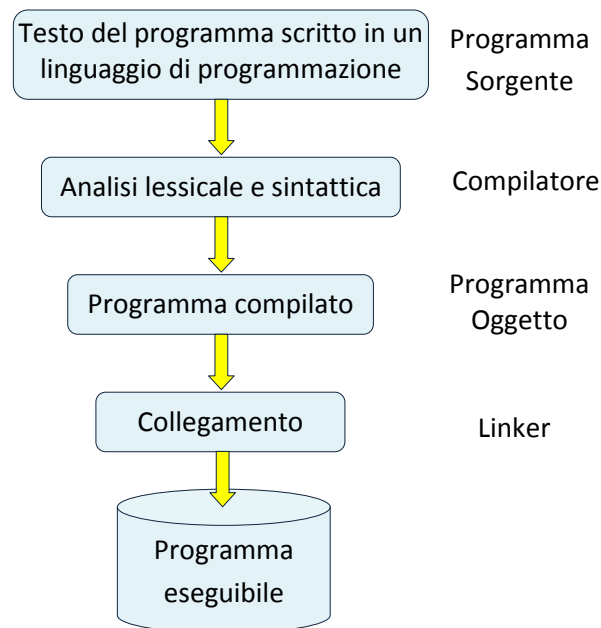
- ✚ il **FORTRAN** (FORmula TRANslator), uno dei primi ad essere usato in particolare per l'esecuzione di calcoli di tipo scientifico;
- ✚ il **COBOL** (Common Business Oriented Language), impiegato per applicazioni di tipo gestionale;
- ✚ il **C**, utilizzato per applicazioni di tipo ingegneristico;
- ✚ il **JAVA**, progettato per applicazioni su Internet;
- ✚ il **PASCAL**, particolarmente adatto per la didattica della programmazione;
- ✚ il **BASIC**, inizialmente progettato per applicazioni scientifiche e didattiche, viene attualmente utilizzato per ogni tipo di problema. Grazie alla facilità con cui viene assimilato e alla grande diffusione dei PC, oggi viene utilizzato nella versione **VISUAL BASIC**.

3.2 Dall'algorithmo al programma

Il linguaggio di programmazione serve per descrivere l'algorithmo risolutivo di un problema in una forma (**programma sorgente**) comprensibile sia dall'uomo sia dall'elaboratore. La persona che è in grado di effettuare tale operazione è chiamato **programmatore**.

Il programma sorgente è poi tradotto dal compilatore (o interprete) in linguaggio macchina. Il programma tradotto in linguaggio macchina è detto **programma oggetto**.

Dopo la compilazione viene effettuata l'operazione di linking o di collegamento (svolta da un programma chiamato linker o collegatore). Tale operazione consiste nell'aggiungere al programma compilato i moduli del compilatore che realizzano le funzioni di basso livello richieste dalle istruzioni del programma. Alla fine di questo procedimento si ottiene il sospirato **programma eseguibile** (.exe).



Il compilatore effettua la compilazione se il programma sorgente è formalmente corretto, cioè che rispetta le regole sintattiche del linguaggio. Ogni volta che ciò non si verifica il compilatore emette un messaggio di errore.

Gli **errori** possono essere di tipo **lessicale** (uso di termini non appartenenti al linguaggio) oppure di tipo **sintattico** (costruzione di frasi non corrette dal punto di vista delle regole grammaticali del linguaggio).

Il compilatore non è in grado di rilevare **errori logici** (cioè riguardanti la correttezza dell'algorithmo)

Il compilatore non è in grado di rilevare errori che possono verificarsi durante l'esecuzione del programma (**runtime error**), sulla base di particolari valori assunti dai dati durante l'elaborazione (Es. divisione per zero).

Alcuni linguaggi di programmazione utilizzano, al posto del programma compilatore, il programma interprete, che traduce il programma sorgente in linguaggio macchina un'istruzione per volta e la esegue. Pertanto, eventuali errori formali (lessicali o sintattici) vengono rilevati e segnalati solo quando l'istruzione errata viene tradotta e causano l'interruzione dell'esecuzione del programma.

3.3 Visual Basic for Application

Per iniziare l'attività di programmazione è possibile utilizzare l'ambiente di programmazione contenuto all'interno del pacchetto Office di Microsoft. In particolare, si può utilizzare il linguaggio di programmazione Visual Basic di Excel. Un primo passo nell'esplorazione di questo ambiente è rappresentato dalla costruzione delle macro.

3.4 Le Macro

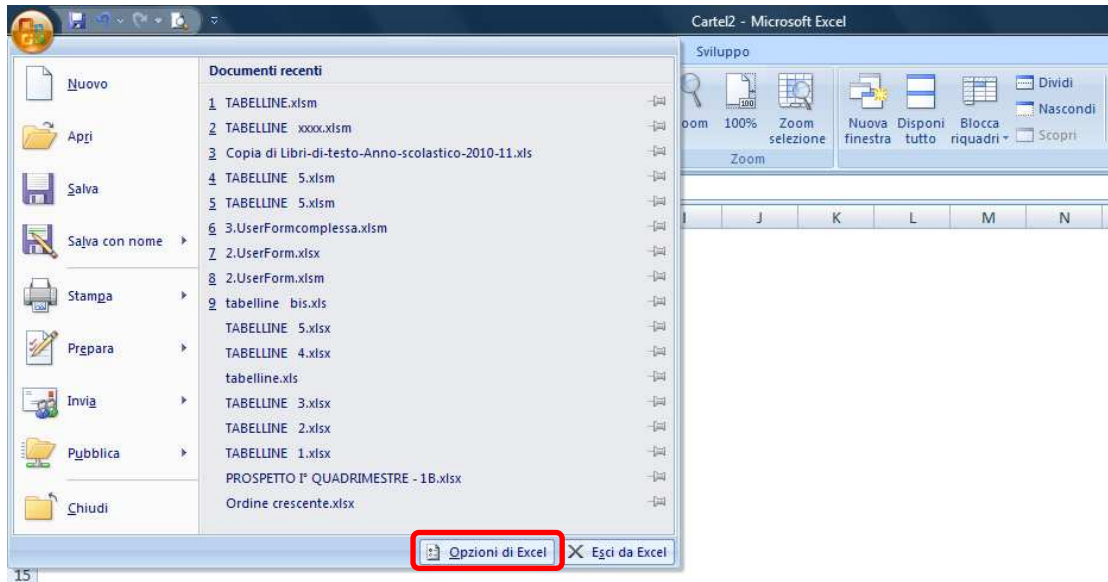
Una **macro** è una sequenza di istruzioni scritte in Visual Basic e organizzate in una subroutine (sottoprogramma).

Essa serve per automatizzare lavori ripetitivi. La registrazione delle macro è molto utile perché permette di vedere la sintassi necessaria per la gestione degli oggetti Excel. Il codice registrato dietro la macro è una valida base di partenza per lo sviluppo di nuovi progetti.

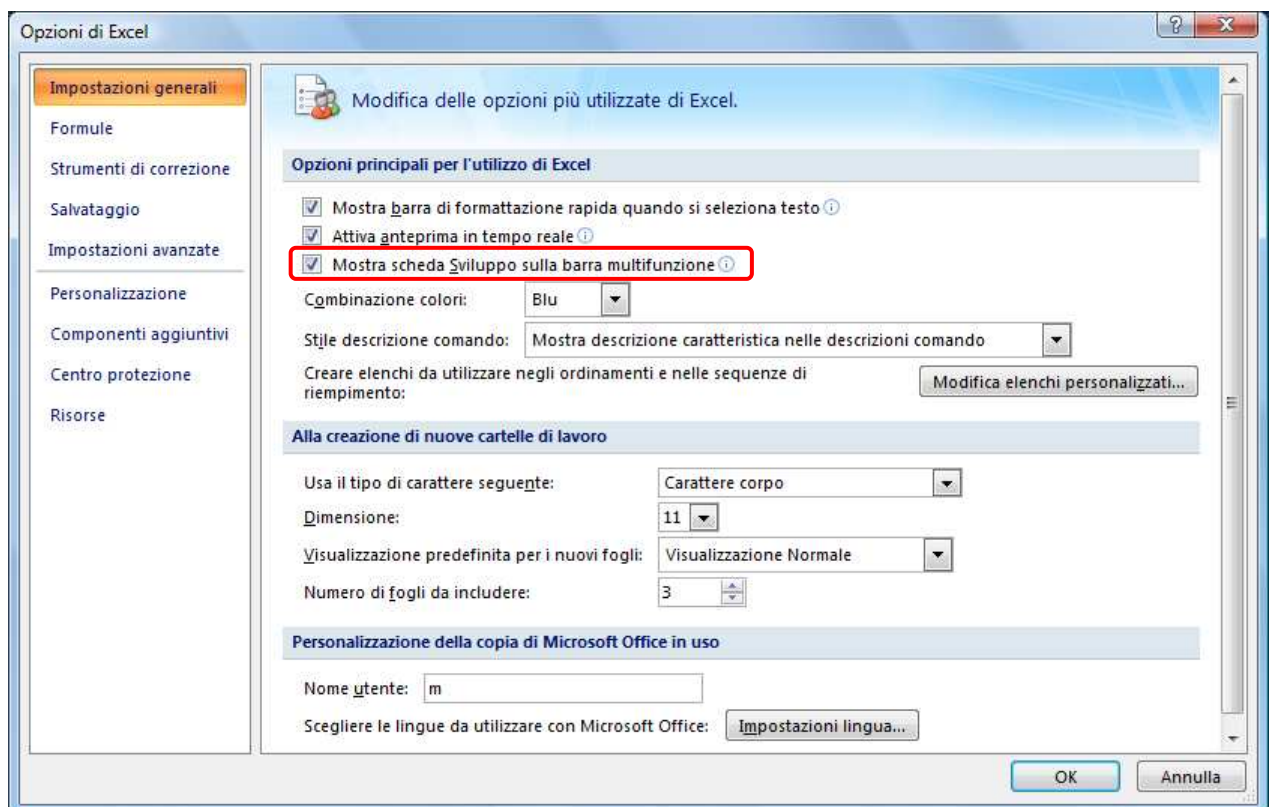
3.4.1 Registrare una macro

Per registrare una macro occorre:

1. Cliccare sul menu **Sviluppo**
2. Se la scheda **Sviluppo** non fosse visibile occorre:
 - a. cliccare il pulsante con il logo di Microsoft Office

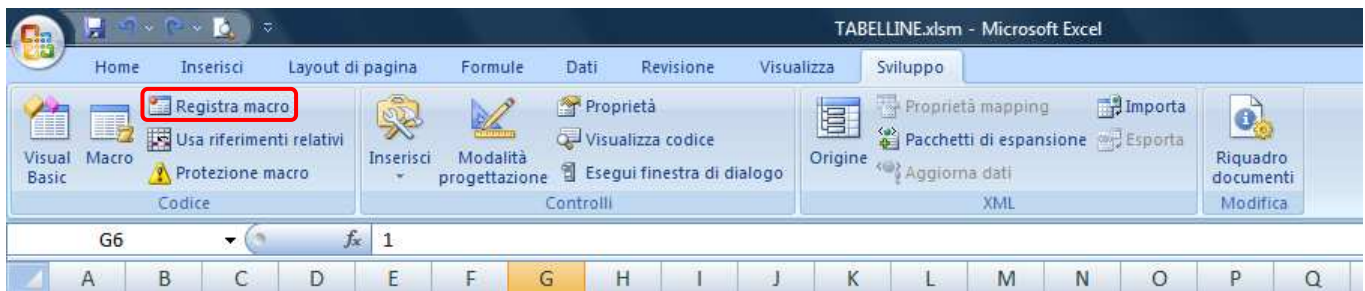


- b. Cliccare il pulsante **Opzioni di Excel**
- c. Nella categoria **Impostazioni generali**, in **Opzioni principali per l'utilizzo di Excel**, selezionare la casella di controllo **Mostra scheda Sviluppo sulla barra multifunzione** e confermare cliccando il pulsante **OK**

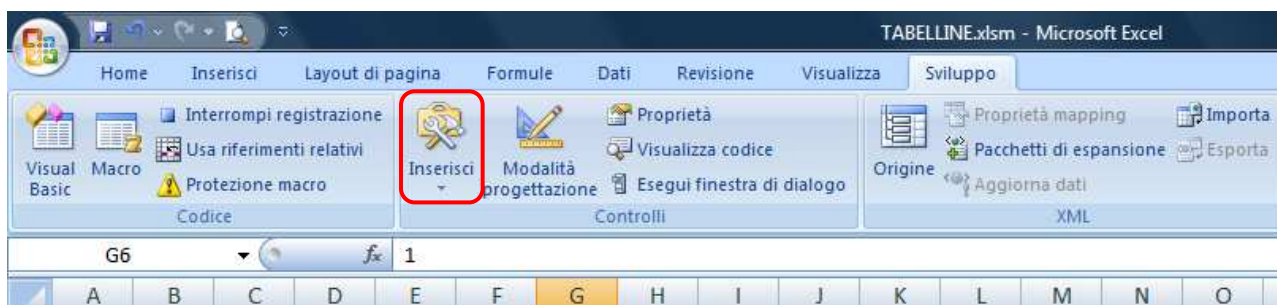


Per registrare una macro occorre quindi:

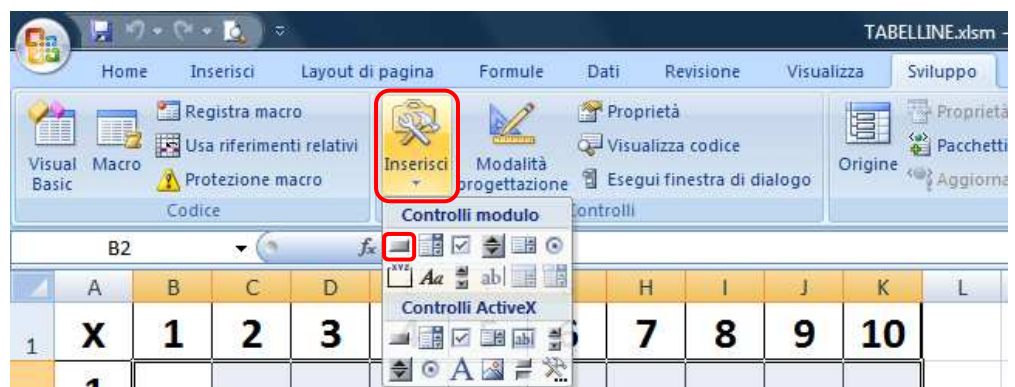
1. Selezionare il menu **Sviluppo**
2. Cliccare il pulsante **Registra macro**



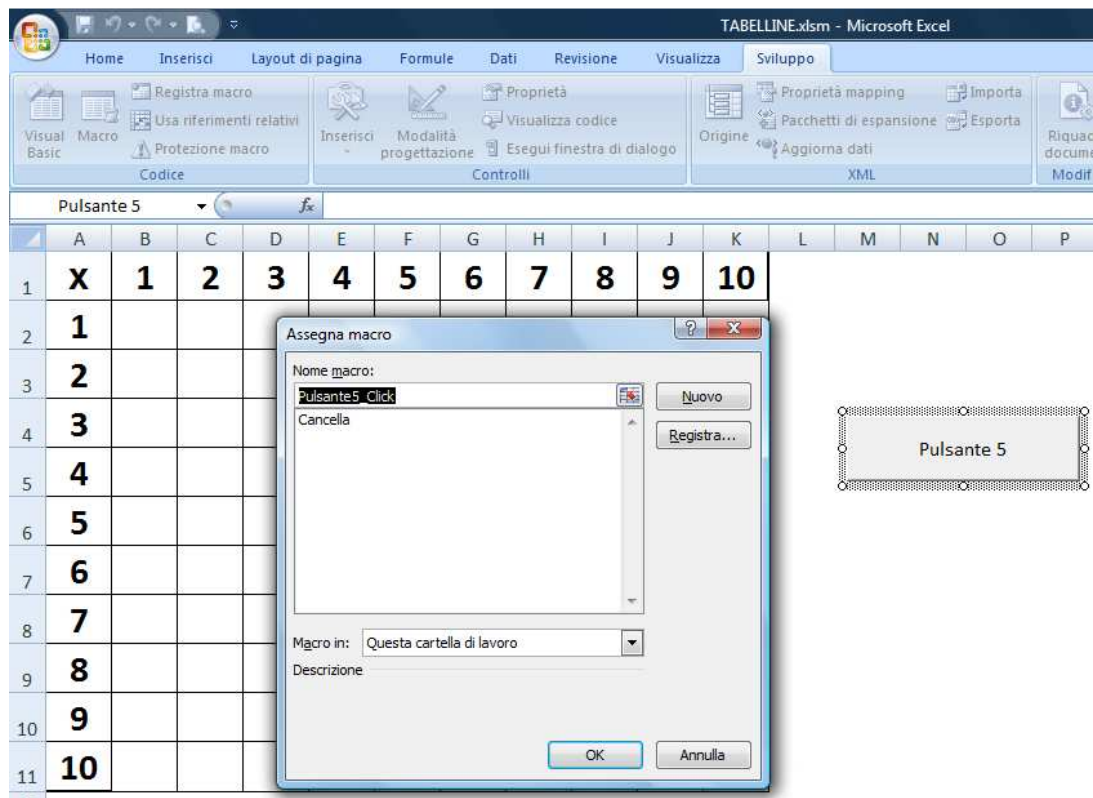
3. Assegnare un nome alla macro (Es. Cancella)
4. Assegnare un tasto di scelta rapida (Es. *Ctrl + X*)
5. Dare una descrizione di ciò che fa la macro
6. Confermare cliccando il pulsante **Ok**
7. Effettuare le operazioni che si intende registrare nella macro
8. Interrompere la registrazione della macro cliccando il pulsante **Interrompi registrazione**
9. Provare il funzionamento della macro cliccando la combinazione di tasti di scelta rapida scelto nel punto 4 (*nel nostro esempio Ctrl + X*)
10. Inserire un pulsante per avviare la macro, cliccando il pulsante **Inserisci** del menu **Sviluppo**



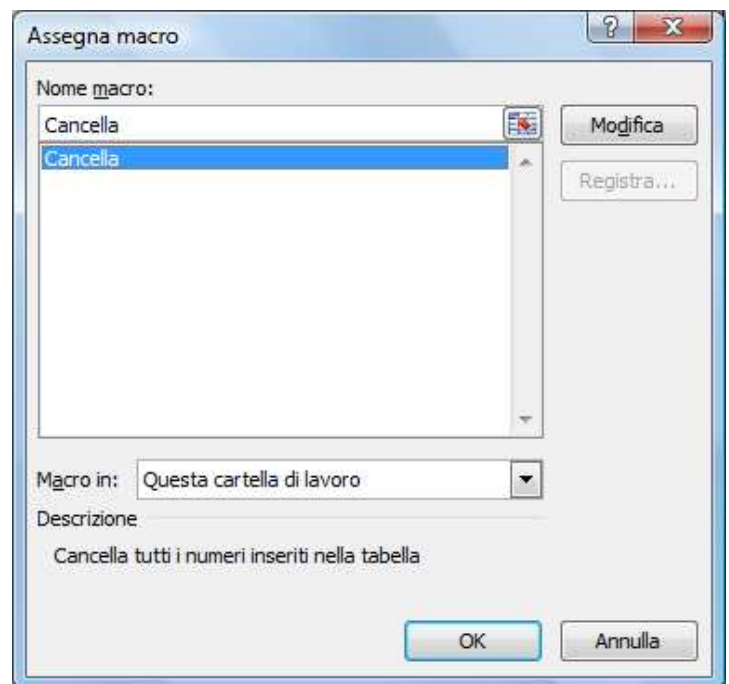
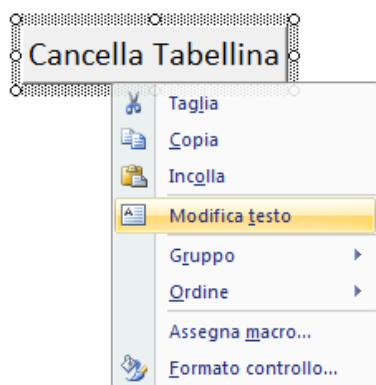
11. Cliccare il **pulsante controllo modulo**



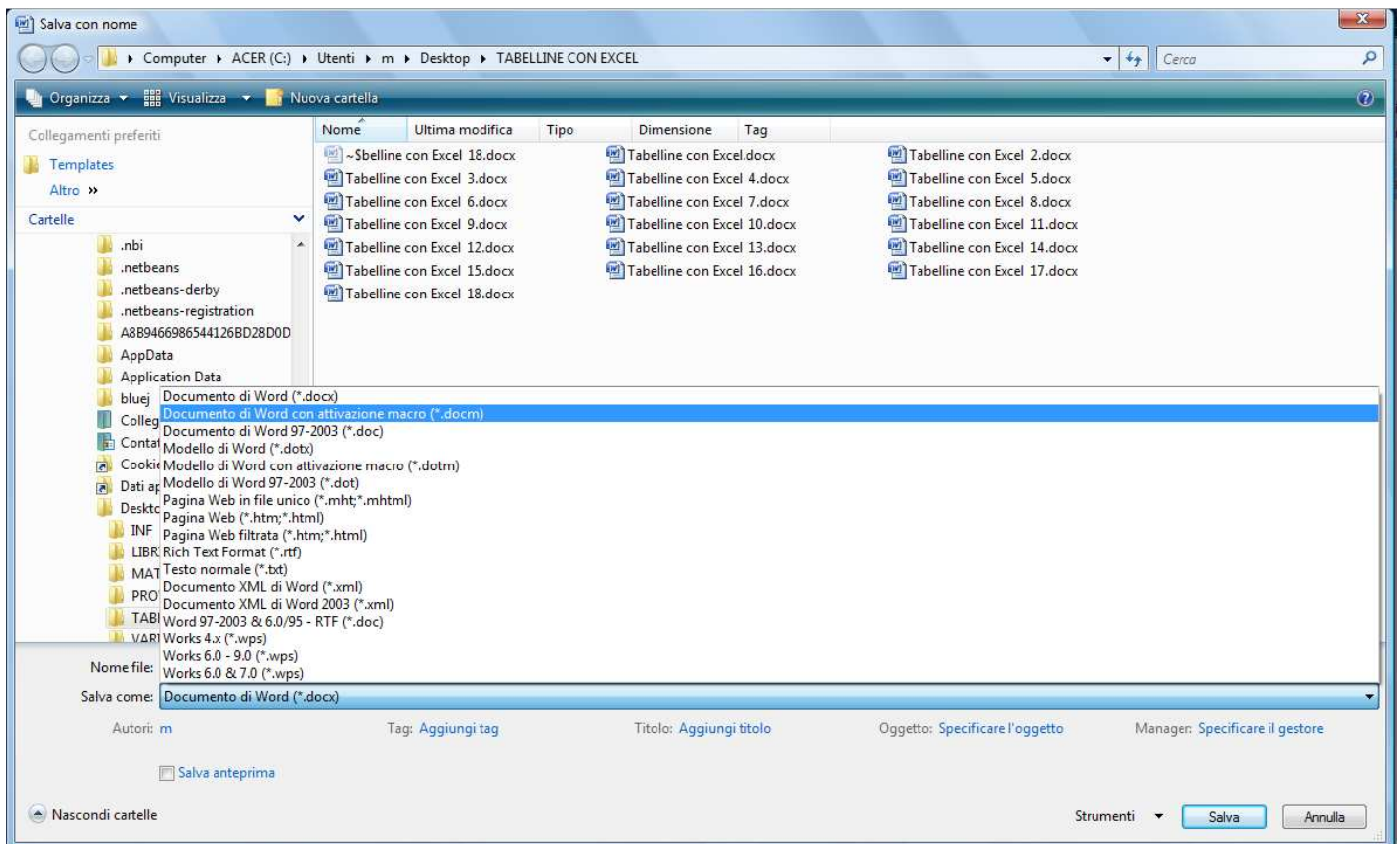
12. Cliccare in un punto libero del foglio e trascinare il mouse per disegnare il pulsante



13. Rilasciare il pulsante del mouse
14. Nella finestra **Assegna macro** che si apre, selezionare la macro (*Nell'esempio "Cancella"*)
15. Cliccare il pulsante **OK**
16. Cliccare, con il tasto destro del mouse, sul pulsante appena creato
17. Cliccare sulla voce **Modifica testo** per modificare l'etichetta del pulsante



18. Salvare il file come **Documento di Word con attivazione macro**

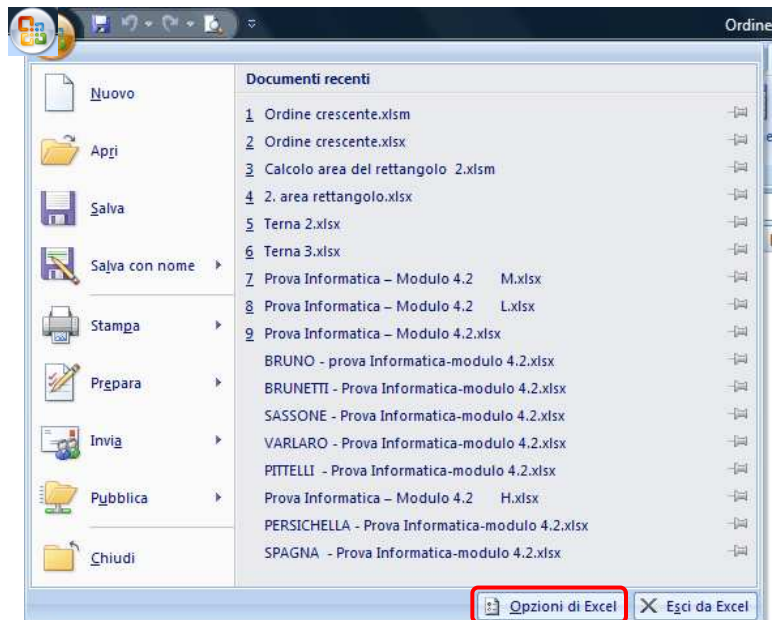


3.4.2 Attivare una macro

Può capitare che le macro non siano attivate, e conseguentemente il pulsante macro appena creato non funzioni.

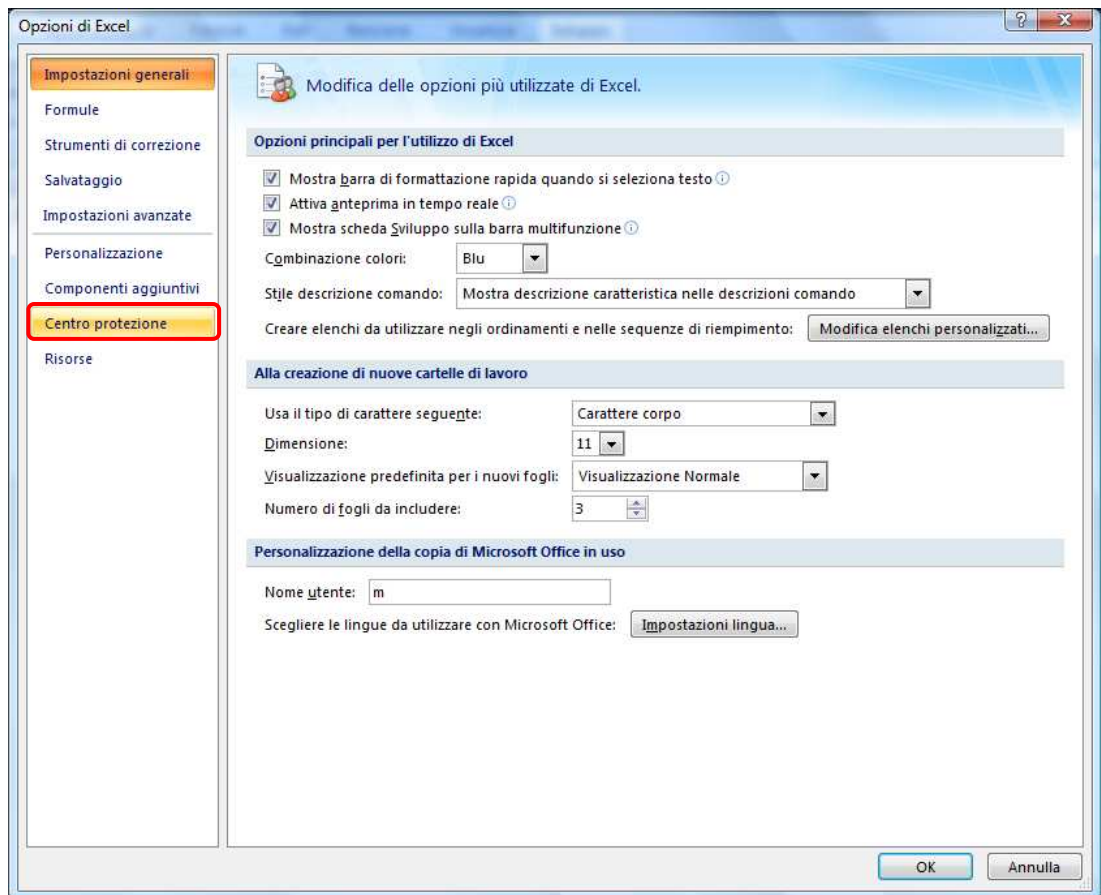
Per attivare le macro occorre:

19. cliccare sul pulsante **Microsoft Office**

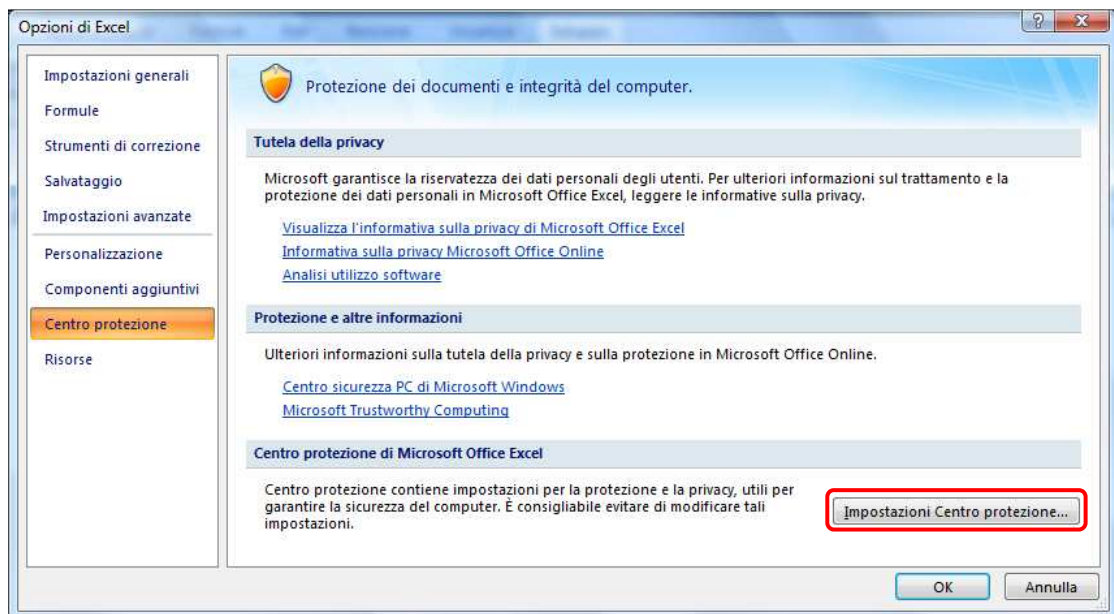


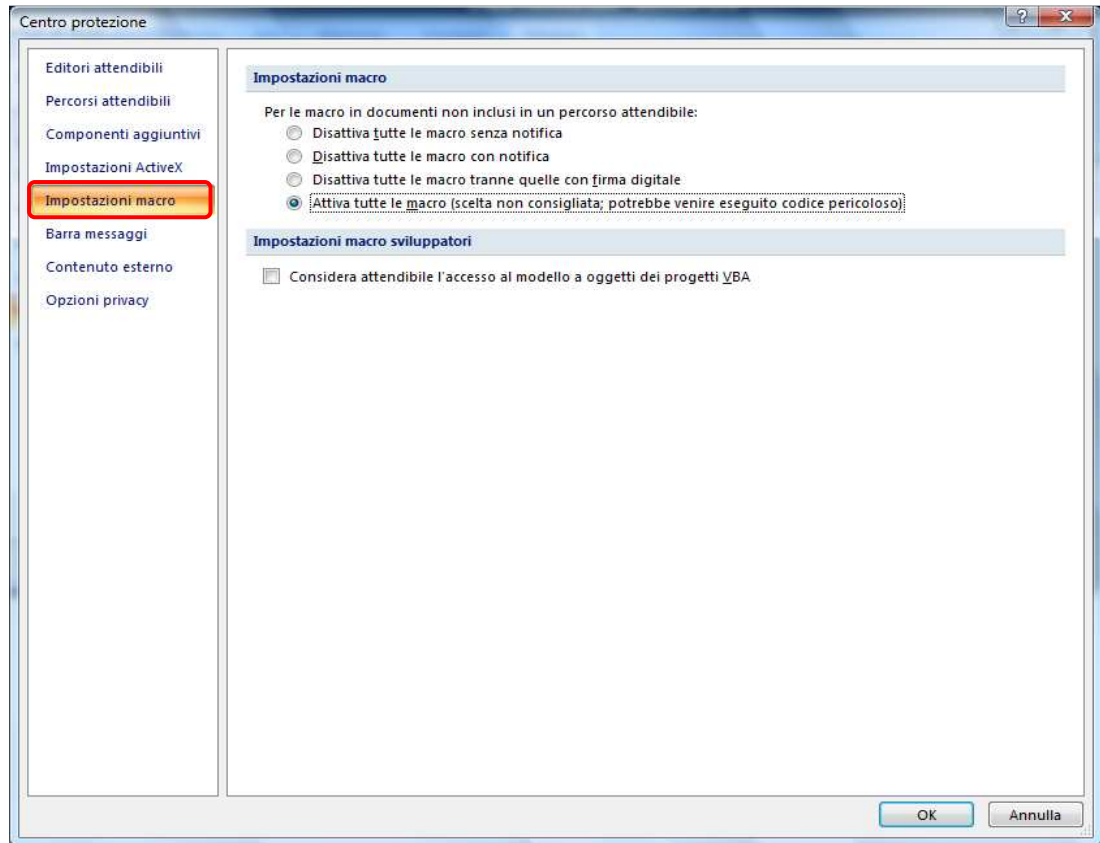
20. cliccare sul pulsante **Opzioni di Excel**

21. cliccare sul pulsante **Centro protezione**



22. cliccare il pulsante **Impostazioni Centro protezione**



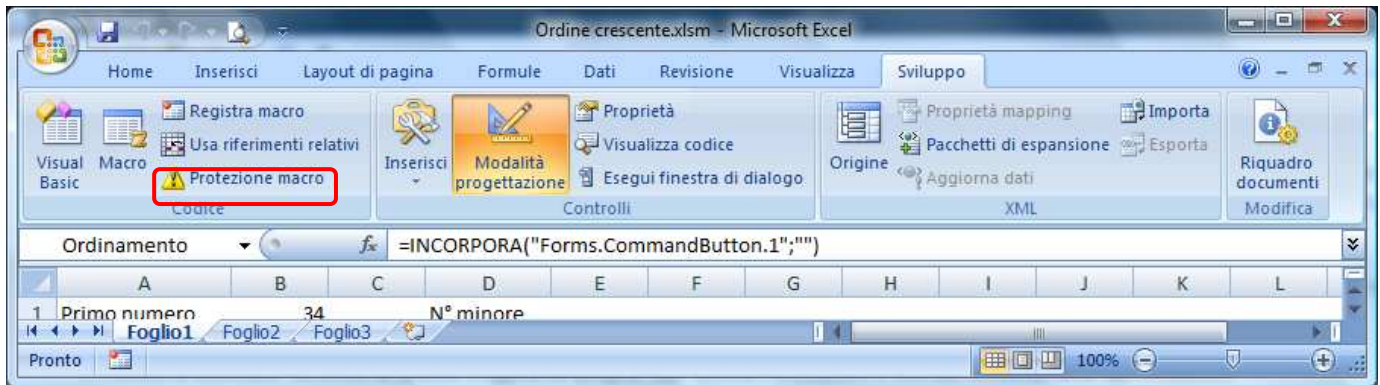


23. Scegliere l'opzione desiderata.

- a. **Disattiva tutte le macro senza notifica** Selezionare questa opzione se le macro non sono considerate attendibili. Tutte le macro contenute nel documento e i relativi avvisi di protezione vengono disattivati. Se esistono documenti con macro non firmate considerate attendibili, è possibile inserire tali documenti in un percorso attendibile. I documenti memorizzati in percorsi attendibili possono essere utilizzati senza alcun controllo del sistema Centro protezione.
- b. **Disattiva tutte le macro con notifica** Impostazione predefinita. Selezionare questa opzione se si desidera che le eventuali macro presenti vengano disattivate visualizzando tuttavia gli avvisi di protezione. In questo modo, è possibile scegliere se attivare tali macro a seconda del caso specifico.
- c. **Disattiva tutte le macro tranne quelle con firma digitale** Questa impostazione è identica all'opzione **Disattiva tutte le macro con notifica** tranne per il fatto che, se la macro include la firma digitale di un editore considerato attendibile, è possibile eseguire la macro a condizione di aver già considerato attendibile l'editore. In caso contrario, non viene visualizzata alcuna notifica. In questo modo, è possibile scegliere se attivare le macro firmate o considerare attendibile l'editore. Tutte le macro non firmate vengono disattivate senza notifica.
- d. **Attiva tutte le macro (scelta non consigliata, potrebbe venire eseguito codice dannoso)** Selezionare questa opzione per consentire l'esecuzione di tutte le macro. Questa impostazione espone il computer al codice potenzialmente dannoso e pertanto non è consigliata.
- e. **Considera attendibile l'accesso al modello a oggetti dei progetti VBA**. Questa impostazione è riservata esclusivamente agli sviluppatori.

Nota

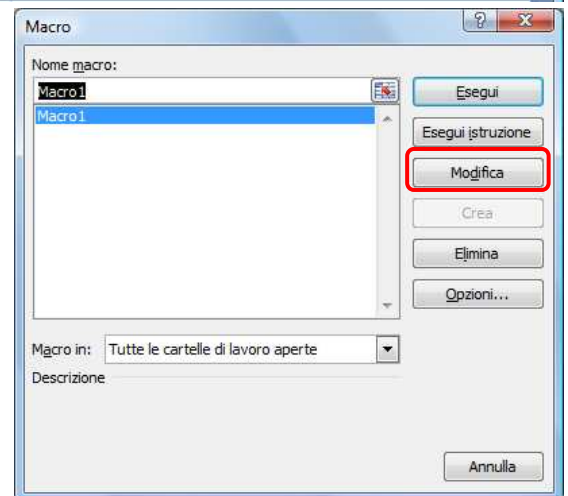
É possibile aprire la finestra di dialogo relativa alle impostazioni di protezione per le macro anche dalla scheda **Sviluppo** sulla barra multifunzione.



3.4.3 Visualizzare il codice Visual Basic associato ad una macro

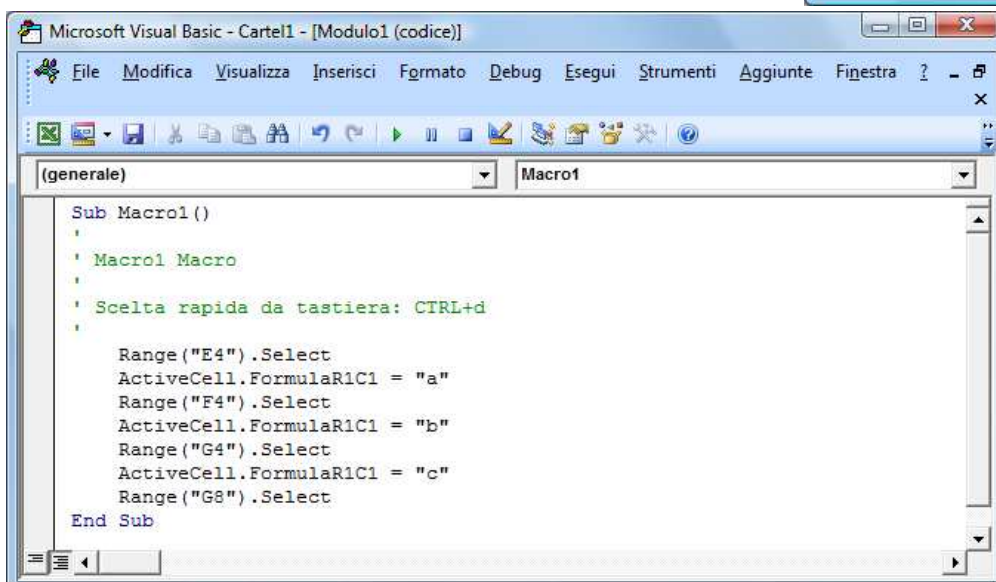
Per visualizzare il codice Visual Basic associato ad una macro occorre:

1. Selezionare il menu **Sviluppo**
2. Cliccare il pulsante **Macro**
3. selezionare il nome della macro
4. cliccare il pulsante **Modifica**.



Come si può osservare, in corrispondenza delle azioni registrate dell'utente, il programma Excel ha creato un sottoprogramma scritto in linguaggio Visual Basic.

Il codice può essere modificato direttamente in questa finestra dall'utente.



3.4.4 Eseguire una macro

Per eseguire una macro occorre:

1. Selezionare il menu **Sviluppo**
2. Cliccare il pulsante **Macro**
3. selezionare il nome della macro
4. cliccare il pulsante **Esegui**

Un'altra modalità di avvio di una macro consiste nell'associare la macro ad un'immagine, oppure a una forma grafica in modo da farla diventare un pulsante di esecuzione macro.

3.4.5 Assegnare una macro

Per assegnare una macro occorre:

1. cliccare con il tasto destro del mouse sull'oggetto grafico
2. selezionare la voce **Assegna Macro...**
3. scegliere la macro desiderata dall'elenco presente nella finestra Macro

Dopo l'assegnazione della macro, passando con il mouse sopra l'oggetto grafico, il puntatore del mouse assume la forma di una piccola mano (collegamento). Facendo clic con il mouse sopra l'oggetto grafico la macro viene eseguita.

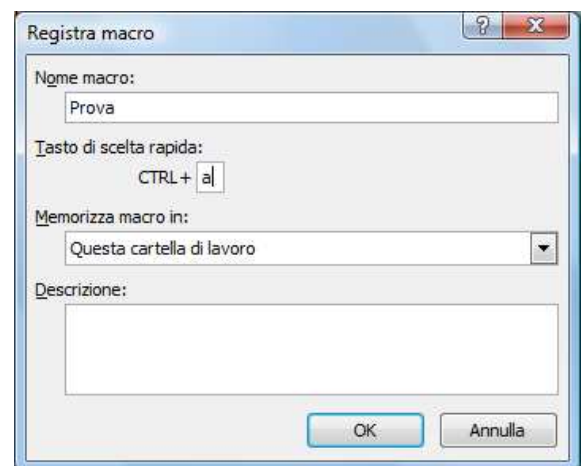
Esempio

Creare una macro che applichi lo sfondo giallo e il colore del carattere blu a tutto il foglio di lavoro.

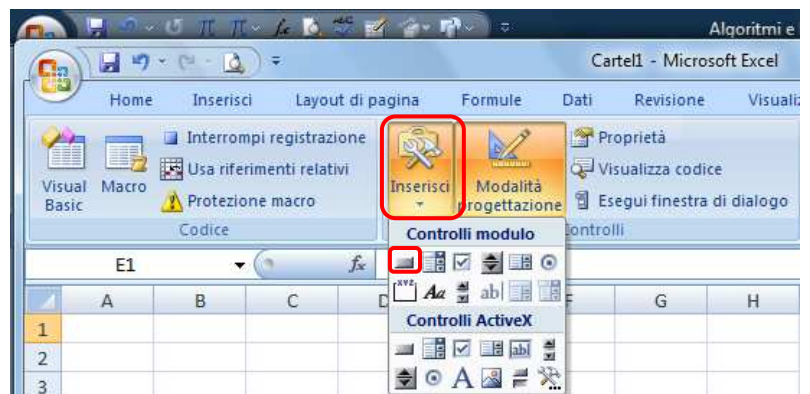
Soluzione

Per creare tale macro occorre:

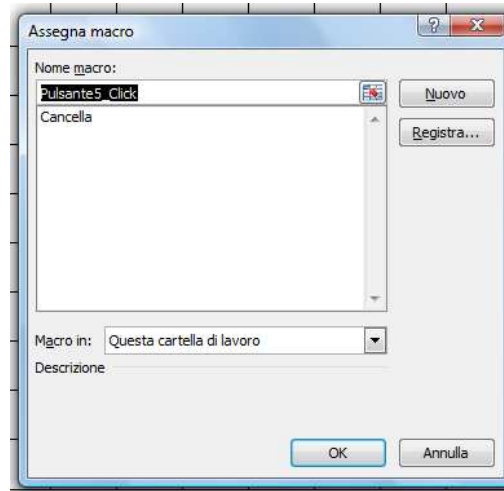
1. Selezionare il menu **Sviluppo**
2. Cliccare il pulsante **Registra Macro**
3. Assegnare un nome alla macro
4. Assegnare un tasto di scelta rapida
5. Cliccare il pulsante **Ok**
6. Selezionare tutto il foglio di lavoro
7. Applicare lo sfondo giallo
8. Applicare il colore blu per il carattere
9. Cliccare il pulsante **Interrompi registrazione**



10. Cliccare il pulsante **Inserisci** nella cassetta degli attrezzi
11. Cliccare il **pulsante controllo modulo**



12. Cliccare in un punto libero del foglio e trascinare il mouse per disegnare un pulsante

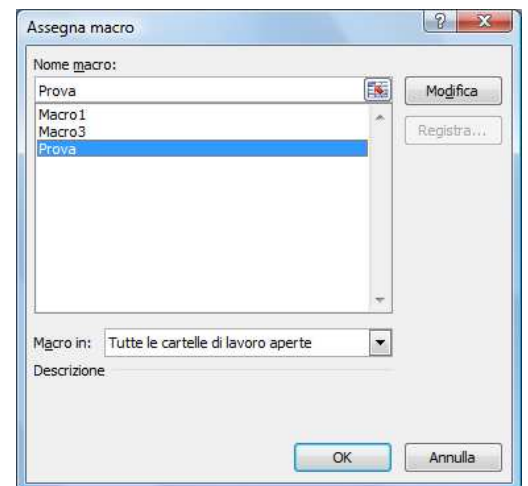
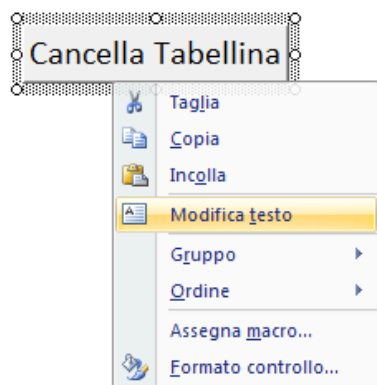


13. Rilasciare il pulsante del mouse

14. Nella finestra **Assegna macro** che si apre, selezionare la macro (Es. Prova)

15. Cliccare il pulsante OK

16. Cliccare, con il tasto destro del mouse, sul pulsante appena creato



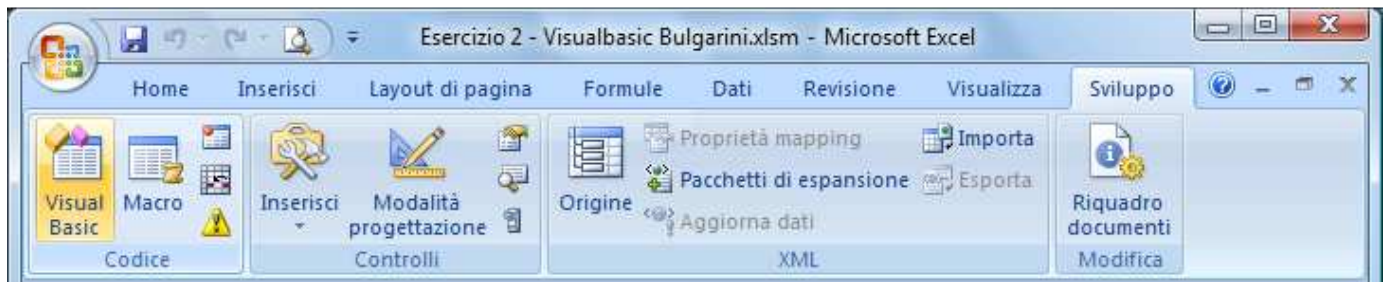
24. Cliccare sulla voce **Modifica testo** per modificare l'etichetta del pulsante

25. Salvare il file come **Documento di Word con attivazione macro**

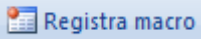

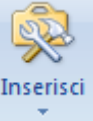
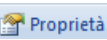

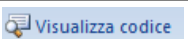

3.5 l'ambiente Visual Basic for application

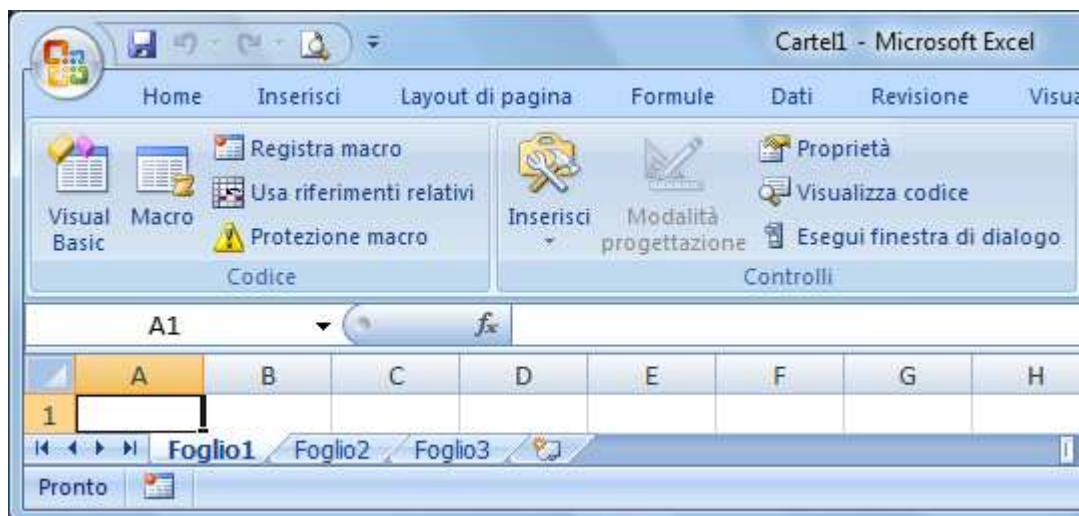
Per utilizzare *Visual Basic* occorre:

1. Cliccare sul menu *Sviluppo*



2. Utilizzare i pulsanti visualizzati nella scheda *Sviluppo*

Pulsante	Funzione
 Registra macro	Per avviare la registrazione di una macro
 Macro	Per aprire il menu di gestione delle macro
 Inserisci	Per inserire gli strumenti grafici per costruire l'interfaccia grafica del progetto: caselle di testo, etichette, pulsanti di comando, barre di scorrimento, liste, ...
 Proprietà	Per aprire la finestra delle proprietà dell'oggetto grafico selezionato
 Visual Basic	Per inserire o modificare il codice Visual Basic del progetto
 Visualizza codice	Per visualizzare il codice Visual Basic del progetto
 Modalità progettazione	Per passare dalla Modalità progettazione alla modalità Esecuzione e viceversa



Le fasi tipiche di lavoro della programmazione in Visual Basic sono:

- ✚ Attivazione della **Modalità progettazione** del progetto cliccando sull'omonimo pulsante
- ✚ Inserimento nel progetto di pulsanti e controlli grafici utilizzando il pulsante **Inserisci** controlli della cassetta degli attrezzi
- ✚ Inserimento del codice Visual Basic associato ai pulsanti e ai controlli grafici effettuando un doppio clic sull'oggetto
- ✚ Disattivazione della **Modalità progettazione**
- ✚ esecuzione del progetto facendo clic sull'oggetto grafico.

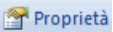
La scorciatoia da tastiera per passare velocemente dal foglio di Excel alla finestra del codice Visual Basic, e viceversa, è la combinazione dei tasti Alt + F11.

3.5.1 Proprietà dei controlli grafici

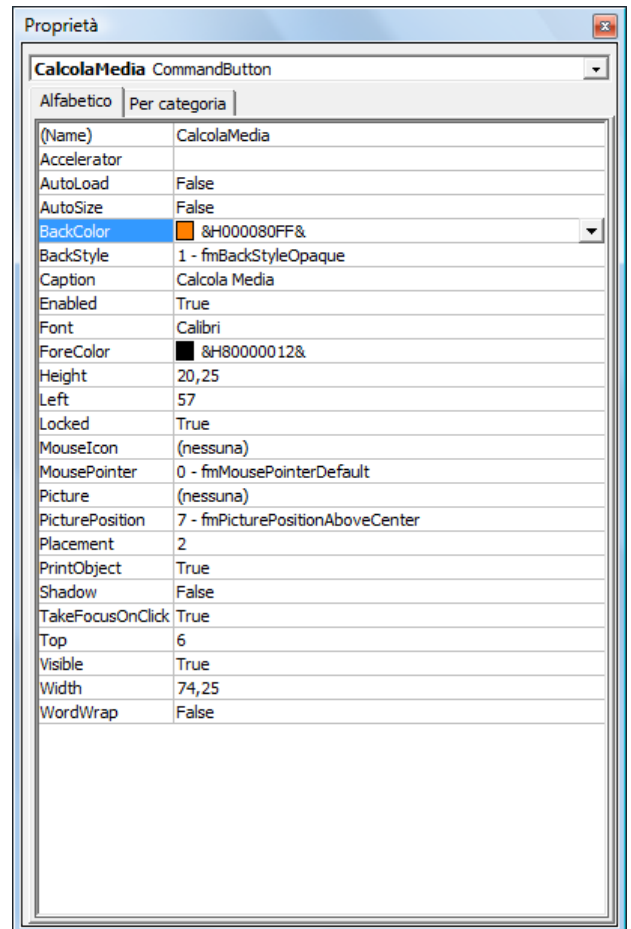
Gli oggetti inseriti nel progetto possiedono alcune proprietà che possono essere personalizzate dal programmatore. Tali proprietà riguardano:

la forma, il colore, il nome, la disposizione dell'oggetto,...

Per modificare le proprietà di un oggetto occorre:

- ✚ cliccare su di esso con il pulsante destro del mouse
- ✚ selezionare nel menu contestuale che si apre la voce **Proprietà** oppure cliccare il pulsante 

Calcola Media



3.5.2 Gli eventi

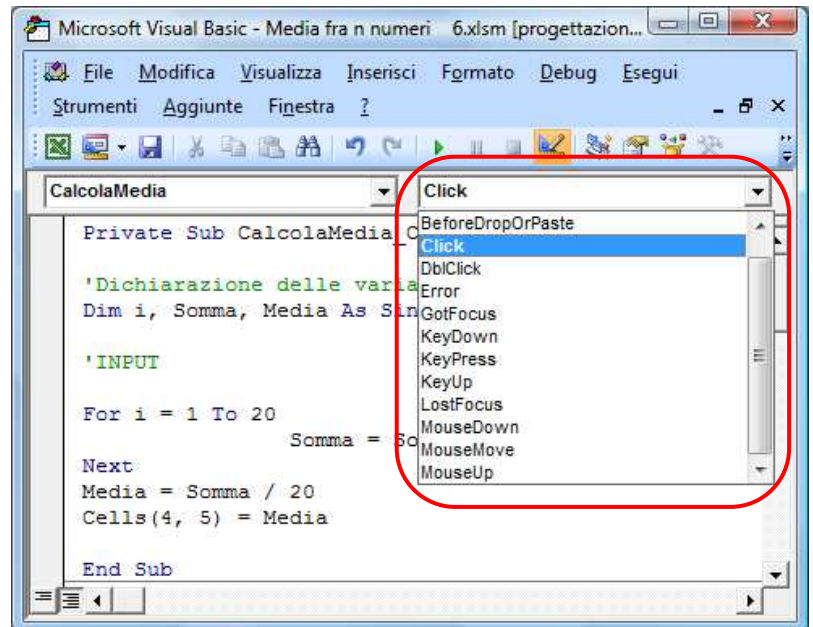
Un evento è un comando impartito dall'utente. Esso può essere effettuato in diversi modi:

- ✚ con un clic sull'oggetto grafico *Pulsante_Click()*
- ✚ con un doppio clic sull'oggetto grafico *Pulsante_DblClick()*
- ✚ con il passaggio del mouse sull'oggetto grafico *Pulsante_MouseMove()* (MouseMove).

I possibili eventi vengono visualizzati facendo clic sulla casella **Click** a destra della finestra del codice.

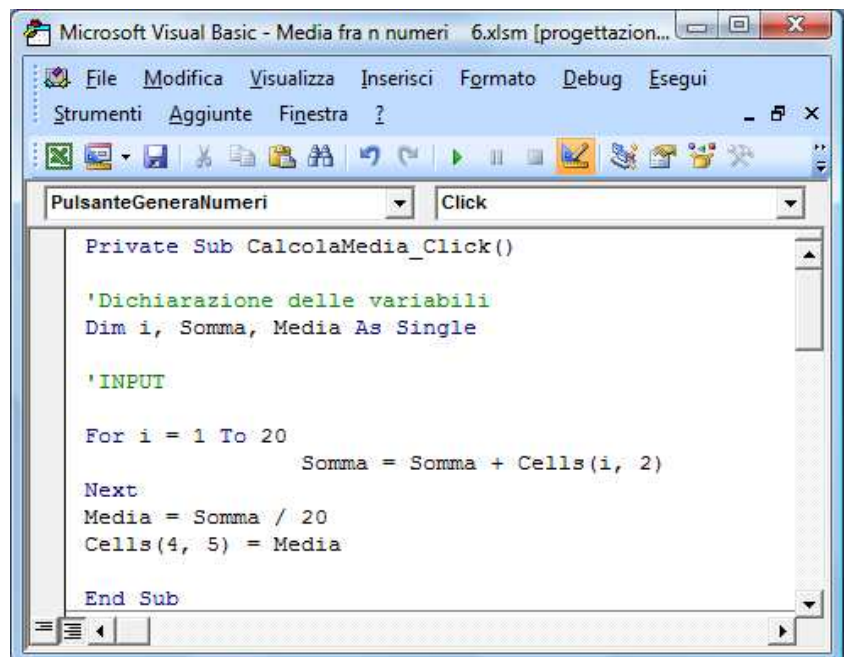
Ad ogni oggetto grafico inserito nel progetto è associato un sottoprogramma (una sequenza di istruzioni)

Il codice associato specifica cosa deve fare il computer come risposta alle azioni dell'utente.



Le istruzioni sono raggruppate tra la frase **Sub** e la frase **End Sub**, per indicare l'inizio e la fine del sottoprogramma (Subroutine) associato all'evento.

L'intestazione del sottoprogramma contiene il nome dell'oggetto grafico (nell'esempio, **CalcolaMedia**) e l'evento da gestire (nell'esempio, **Click**).



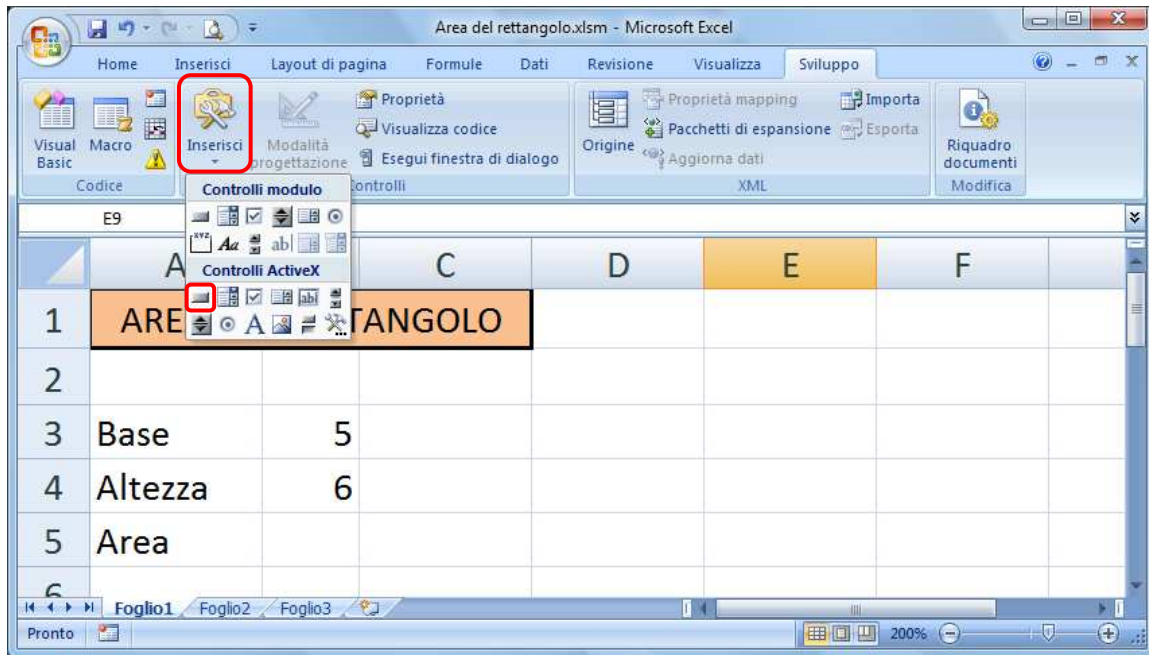
3.5.3 Struttura di sequenza

Il problema seguente presenta un esempio di codifica in Visual Basic della struttura di sequenza vista al paragrafo 2.2. Dati la misura della base b e dell'altezza h di un rettangolo, calcola l'area del rettangolo S .

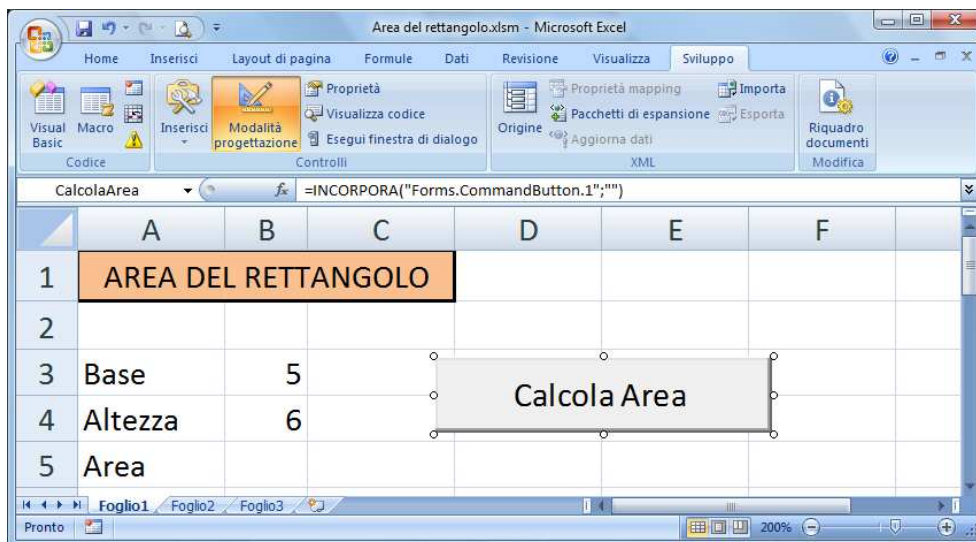
Soluzione

Per costruire il foglio elettronico che risolve tale problema occorre:

1. Aprire *Microsoft Excel*
2. Salvare il file con il *nome* *Area del rettangolo* e *Salva come* Cartella di lavoro con attivazione di Macro di Excel
3. Creare la seguente tabella

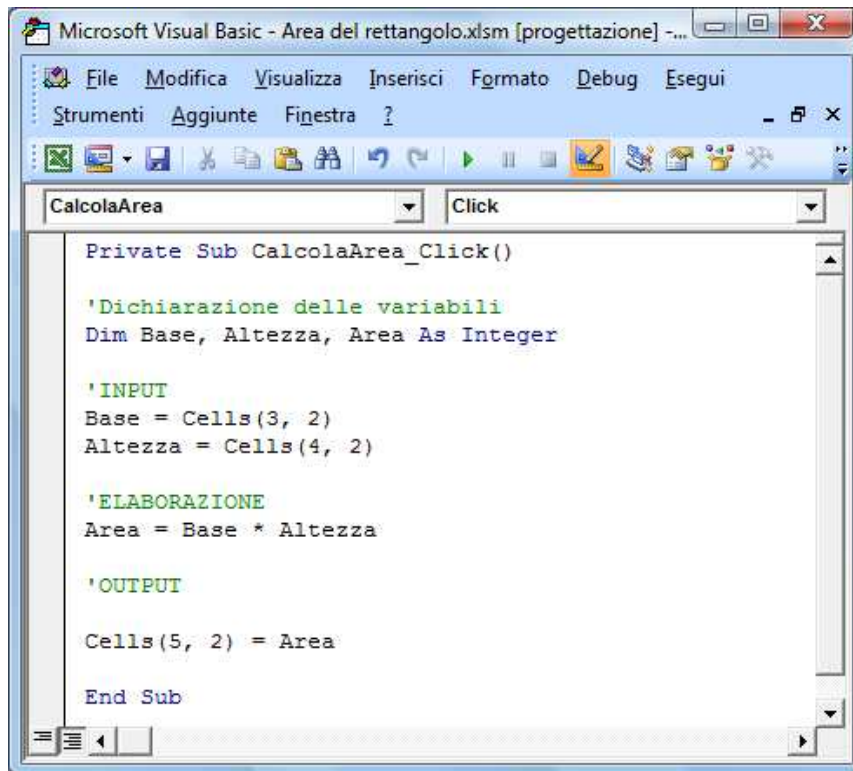


4. Cliccare sul pulsante *Inserisci* del menu *Sviluppo* per inserire un *pulsante di comando* per consentire all'utente di attivare il codice



5. cliccare con il pulsante destro del mouse sul pulsante creato e scegliere, dal menu di scelta rapida che si apre, la voce *Proprietà*
6. cambiare la proprietà *Caption*, che rappresenta il testo che compare sopra il pulsante di comando con la frase: *Calcola Area*.

7. Aprire la finestra di Visual Basic (*basta effettuare un doppio clic sul pulsante*) per associare al pulsante il codice del sottoprogramma che deve essere eseguito quando l'utente fa clic sul pulsante (Il codice da inserire è rappresentato in figura)



```

Microsoft Visual Basic - Area del rettangolo.xlsm [progettazione] - ...
File Modifica Visualizza Inserisci Formato Debug Esegui
Strumenti Aggiunte Finestra ?
CalcolaArea Click
Private Sub CalcolaArea_Click()
'Dichiarazione delle variabili
Dim Base, Altezza, Area As Integer

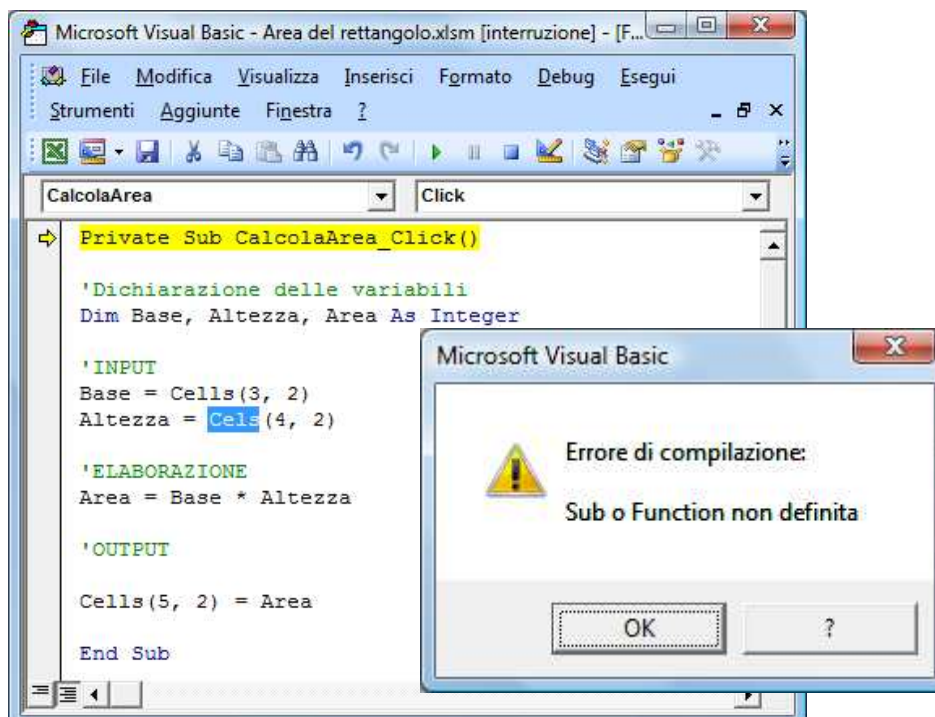
'INPUT
Base = Cells(3, 2)
Altezza = Cells(4, 2)

'ELABORAZIONE
Area = Base * Altezza

'OUTPUT
Cells(5, 2) = Area
End Sub

```

8. Dopo aver inserito il codice chiudere la finestra di Visual Basic
9. Uscire dalla Modalità progettazione cliccando l'omonimo pulsante
10. Cliccare il pulsante Calcola Area per eseguire il sottoprogramma
11. Se non sono stati commessi errori di digitazione compare il risultato dell'area nella cella (5, 2). Altrimenti si apre la finestra di Visual Basic con la segnalazione dell'errore



3.5.4 Il codice

Nel codice del sottoprogramma per il calcolo dell'area del rettangolo, le istruzioni sono mostrate con colori diversi.

Il colore verde è attribuito ai commenti (Per inserire un commento basta farlo precedere dal simbolo di apostrofo)

Il colore blu è attribuito alle parole chiavi

Il colore nero è attribuito alle istruzioni

Il colore rosso è attribuito agli errori

PAROLE CHIAVI		
Parola chiave	Significato	Esempio
Private Sub	Dichiarazione del nome del sottoprogramma	Private Sub CalcolaArea_Click()
Dim	Dichiarazione dei nomi delle variabili utilizzate nel sottoprogramma	Dim Base, Altezza As Integer
Integer	Tipo di dato Intero (- 32 768 , + 32 767)	Dim Altezza As Integer
Cells (riga, colonna)	Individua una cella di Excel	Cells (3, 2)
Range ("ColonnaRiga")	Individua una cella di Excel	Range ("B2")
InputBox ("Prompt", "Title")	Input dati tramite finestra di Windows	InputBox ("Base = ", "Area Rettangolo")
MsgBox Variabile, vbOKCancel, "Testo"	Output dati tramite finestra di Windows	MsgBox Area, vbOKCancel, "L'area è "
=	Operazione di assegnazione	Base = Cells(3, 2) <i>(Assegna alla variabile Base il contenuto della cella (3, 2))</i>
End Sub	Dichiarazione della fine del sottoprogramma	End Sub

OPERATORI MATEMATICI		
Operatori matematici	Significato	Esempio
+	Addizione	$A = B + C$
-	Sottrazione	$A = B - C$
*	Moltiplicazione	$A = B * C$
/	Divisione	$A = B / C$
^	Potenza	$A = 3 ^ 2$ ($3^2 = 9$)
Sqr	Radice quadrata	Sqr (9) (<i>Sqr</i> (9) = 3)
\	Divisione intera	$A = 19 \setminus 5$ ($18 \setminus 5 = 3$)
Mod	Resto della divisione	$A = 19 \text{ Mod } 5$ ($19 \text{ Mod } 5 = 4$)

I TIPI DI DATI			
Sintassi	Tipo	Minimo valore	Massimo valore
Byte	Numero intero a 1 byte	0	255
Integer	Numero intero a 2 byte	- 32 768	+ 32 767
Long	Numero intero a 4 byte	- 2 147 483 648	+ 2 147 483 647
Single	Numero reale a virgola mobile a precisione singola a 2 byte	-3,402823 · 10 ³⁸ tra -1,401298 · 10 ⁻⁴⁵ per valori negativi	
		1,401298 · 10 ⁻⁴⁵ tra 3,402823 · 10 ³⁸ per valori positivi	
Double	Numero reale a virgola mobile a doppia precisione a 8 byte	-1,79769313486232 · 10 ³⁰⁸ tra -4,94065645841247 · 10 ⁻³²⁴ per valori negativi	
		4,94065645841247 · 10 ⁻³²⁴ tra 1,79769313486232 · 10 ³⁰⁸ per valori positivi	
Date	Date e orari	Date dal 1 gennaio 100 al 31 dicembre 9999	
String	Caratteri alfanumerici	Accetta da 0 a 65000 caratteri	
Boolean	Dati booleani	Vero o Falso oppure True o False	

3.5.5 La struttura di selezione

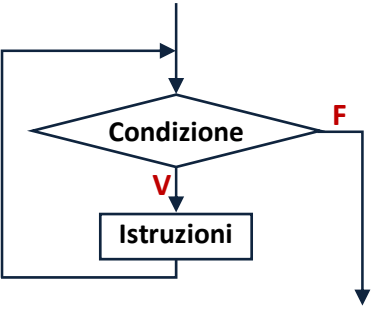
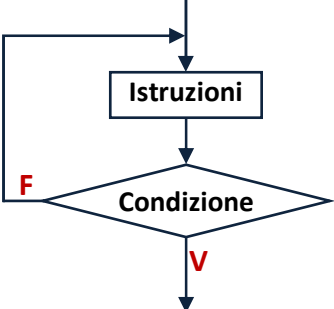
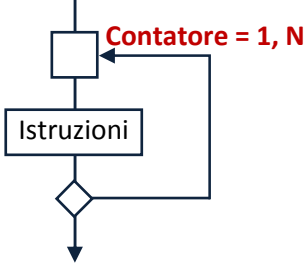
La codifica in linguaggio Visual Basic della struttura di selezione è la seguente:

Codifica in linguaggio Visual Basic	
<i>Selezione a due uscite</i>	<i>Selezione a una uscita</i>
IF Condizione = Vera THEN Istruzione A ELSE Istruzione B END IF	IF Condizione = Vera THEN Istruzione A END IF

3.5.6 La struttura di iterazione

La codifica in linguaggio Visual Basic della struttura di iterazione è la seguente:

Codifica in linguaggio Visual Basic		
<i>Iterazione indefinita precondizionale</i>	<i>Iterazione indefinita postcondizionale</i>	<i>Iterazione definita enumerativa</i>
Do While Condizione = Vera Istruzioni Loop	Do Istruzioni Loop Until Condizione = Falsa	For contatore = Iniziale To Finale Istruzioni Next

Strutture di Iterazione		
<i>Iterazione indefinita precondizionale</i>	<i>Iterazione indefinita postcondizionale</i>	<i>Iterazione definita enumerativa</i>
		
Mentre <i>Condizione = Vera</i> esegui Istruzioni	Ripeti Istruzioni finchè <i>Condizione = Falsa</i>	Ripeti Istruzioni N volte
Esce dal ciclo quando <i>Condizione = Falsa</i>	Esce dal ciclo quando <i>Condizione = Vera</i>	Esce dal ciclo quando <i>Contatore = N</i>

Esempio 1

Calcolare l'ipotenusa di un triangolo rettangolo, noti i cateti (*utilizzando le celle di Excel e l'istruzione Range*)

```
Private Sub Bottonelpotenusa_Click()  
    'Dichiarazione delle variabili  
    Dim c1, c2, ipot As Single  
    'Acquisizione delle misure dei cateti  
    c1 = Range("b4")  
    c2 = Range("b5")  
    'Calcolo dell'ipotenusa  
    ipot = Sqr(c1 ^ 2 + c2 ^ 2)  
    'Visualizzazione del risultato  
    Range("b6") = ipot  
End Sub
```

Esempio 2

Calcolare l'ipotenusa di un triangolo rettangolo, noti i cateti (*le celle di Excel e l'istruzione Cells*)

```
Private Sub Bottonelpotenusa_Click()  
    'Dichiarazione delle variabili  
    Dim c1, c2, ipot As Single  
    'Acquisizione delle misure dei cateti  
    c1 = Cells(4, 2)  
    c2 = Cells(5, 2)  
    'Calcolo dell'ipotenusa  
    ipot = Sqr(c1 ^ 2 + c2 ^ 2)  
    'Visualizzazione del risultato  
    Cells(6, 2) = ipot  
End Sub
```

Esempio 3

Calcolare l'ipotenusa di un triangolo rettangolo, noti i cateti (*utilizzando le finestre di Windows*)

```
Private Sub Bottonelpotenusa_Click()  
    'Dichiarazione delle variabili  
    Dim c1, c2, ipot As Single  
    'Acquisizione delle misure dei cateti  
    c1 = InputBox("Cateto a", "Input Cateti")  
    c2 = InputBox("Cateto b", "Input Cateti")  
    'Calcolo dell'ipotenusa  
    ipot = Sqr(c1 ^ 2 + c2 ^ 2)  
    'Visualizzazione del risultato  
    MsgBox ipot, vbOKCancel, "Output Ipotenusa"  
End Sub
```

Esempio 4

Calcolare il valore assoluto di un numero (utilizzando le celle di Excel e l'istruzione Cells)

```
Private Sub ValoreAssoluto_Click()  
'Dichiarazione della variabile di tipo Single  
Dim A As Single  
'INPUT  
A = Cells(3, 2)  
'ELABORAZIONE  
If A < 0 Then  
    A = -A  
End If  
'OUTPUT  
Cells(4, 2) = A  
End Sub
```

Esempio 5

Calcolare il valore assoluto di un numero (utilizzando le celle di Excel e l'istruzione Range)

```
Private Sub ValoreAssoluto_Click()  
'Dichiarazione della variabile di tipo Single  
Dim A As Single  
'INPUT  
A = Range("B3")  
'ELABORAZIONE  
If A < 0 Then  
    A = -A  
End If  
'OUTPUT  
Range("B4") = A  
End Sub
```

Esempio 6

Calcolare il valore assoluto di un numero (utilizzando le finestre di Windows)

```
Private Sub ValoreAssoluto_Click()  
Dim Numero(1 To 100) As Single  
'INPUT  
Numero(1) = InputBox("Numero A", "Valore Assoluto")  
'ELABORAZIONE  
If Numero(1) < 0 Then  
    Numero(1) = -Numero(1)  
End If  
'OUTPUT  
MsgBox Numero(1), vbOKCancel, "Il valore assoluto di A è :"  
End Sub
```

Esempio 7

Calcola la somma di due frazioni (*utilizzando le celle di Excel e l'istruzione Cells*)

```
Private Sub SommaFrazioni_Click()  
Dim A, B, C, D, NUM, DEN As Integer  
A = Cells(5, 2)  
B = Cells(6, 2)  
C = Cells(5, 4)  
D = Cells(6, 4)  
If B = 0 Then  
    MsgBox "il denominatore non può essere nullo", vbCritical  
    A = 1  
    Cells(6, 2) = A  
End If  
If D = 0 Then  
    MsgBox "il denominatore non può essere nullo", vbCritical  
    D = 1  
    Cells(6, 4) = D  
End If  
DEN = B * D  
NUM = A * D + B * C  
Cells(5, 6) = NUM  
Cells(6, 6) = DEN  
End Sub
```

Esempio 8

Calcola i numeri di Fibonacci (*utilizzando le celle di Excel e l'istruzione Cells*)

```
Private Sub Fibonacci_Click()  
Dim Numero, NumeroFibonacci, NumeroPrecedente, NumeroSuccessivo, ScambioNumeri, Contatore As Integer  
Numero = Cells(3, 2)  
If (Numero = 1) Or (Numero = 2) Then  
    Cells(4, 2) = 1  
Else  
    NumeroPrecedente = 1  
    For Contatore = 1 To Numero  
        ScambioNumeri = NumeroSuccessivo  
        NumeroSuccessivo = NumeroSuccessivo + NumeroPrecedente  
        NumeroPrecedente = ScambioNumeri  
    Next Contatore  
    NumeroFibonacci = NumeroSuccessivo  
    Cells(4, 2) = NumeroFibonacci  
End If  
End Sub
```