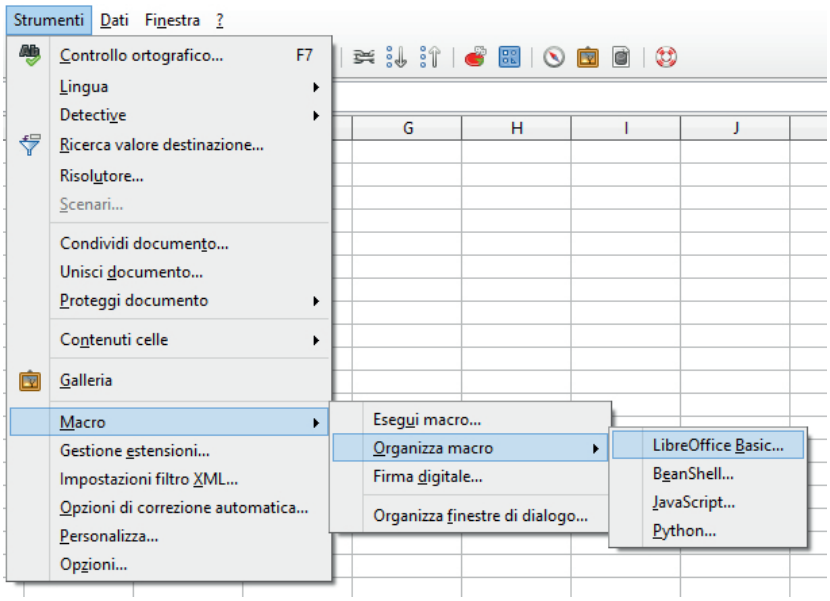
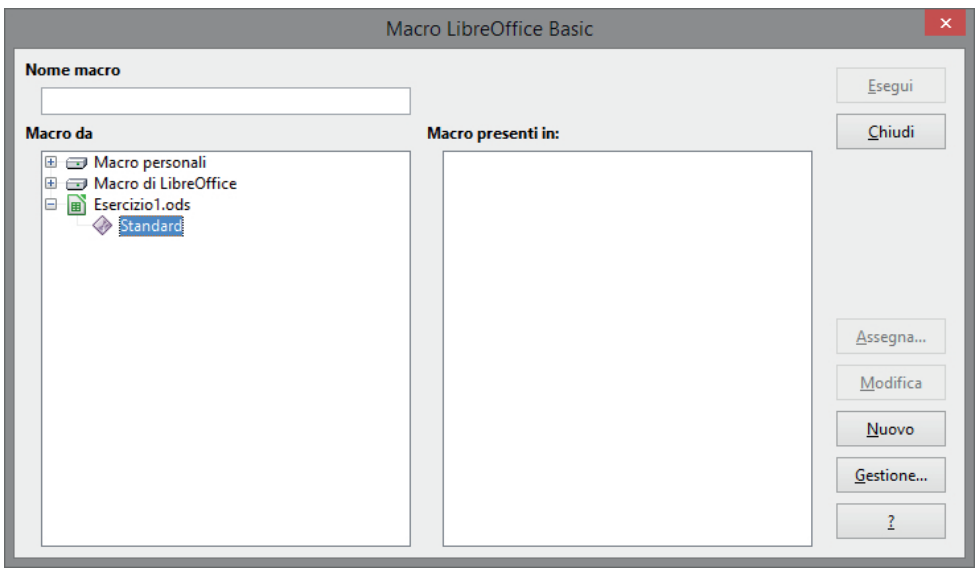
Programmazione con il linguaggio LibreOffice Basic

L'ambiente di programmazione

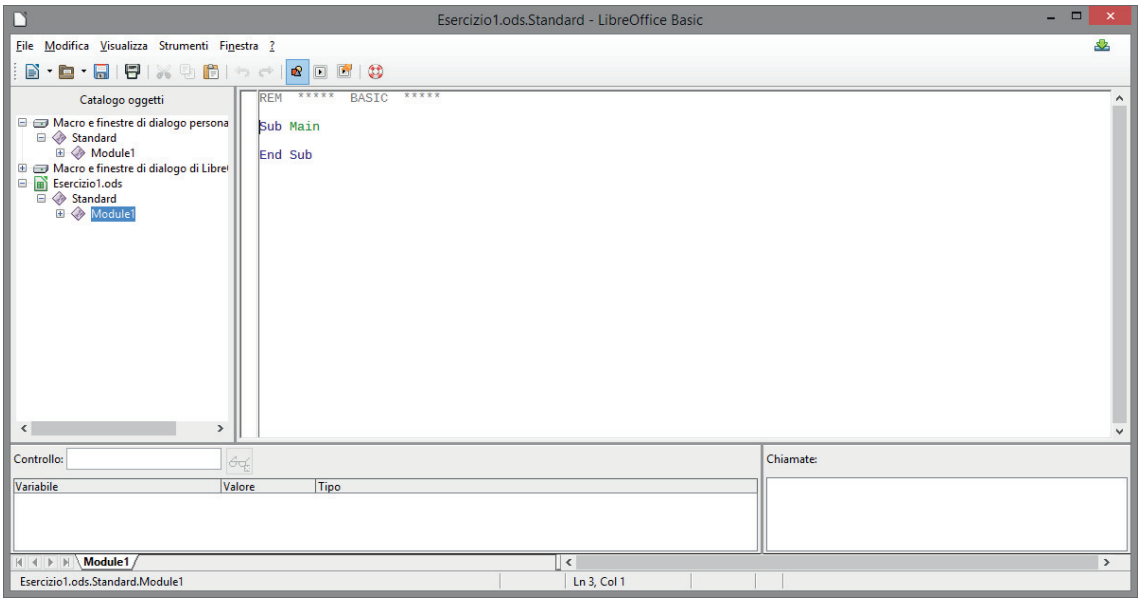
Il software *LibreOffice* possiede un ambiente di programmazione in linguaggio **Basic**, che consente di creare procedure software dalla fase di editing del testo sorgente fino all'esecuzione del programma.
Per attivare questo ambiente, aprire un programma *LibreOffice*, per esempio *Calc:* dal menu **Strumenti**, scelta **Macro, Organizza macro**, **LibreOffice Basic**.



Si apre la finestra delle **macro**: clic sul nome del file *.ods* e poi su **Standard**.
Creare un nuovo **Modulo**, per contenere i sottoprogrammi associati al foglio elettronico, facendo clic sul pulsante **Nuovo**.



Automaticamente si apre l'**editor** di Basic e viene creata la struttura di un sottoprogramma **Main**. Le istruzioni sono raggruppate tra la frase **Sub Main** e la frase **End Sub**, per indicare l'inizio e la fine del sottoprogramma (*Subroutine*).



Istruzioni in sequenza

Il problema seguente presenta un esempio di codifica in Basic della **sequenza**, cioè la struttura dell'algoritmo nella quale le istruzioni sono indicate una di seguito all'altra secondo l'ordine con il quale devono essere eseguite dal computer.

ESEMPIO

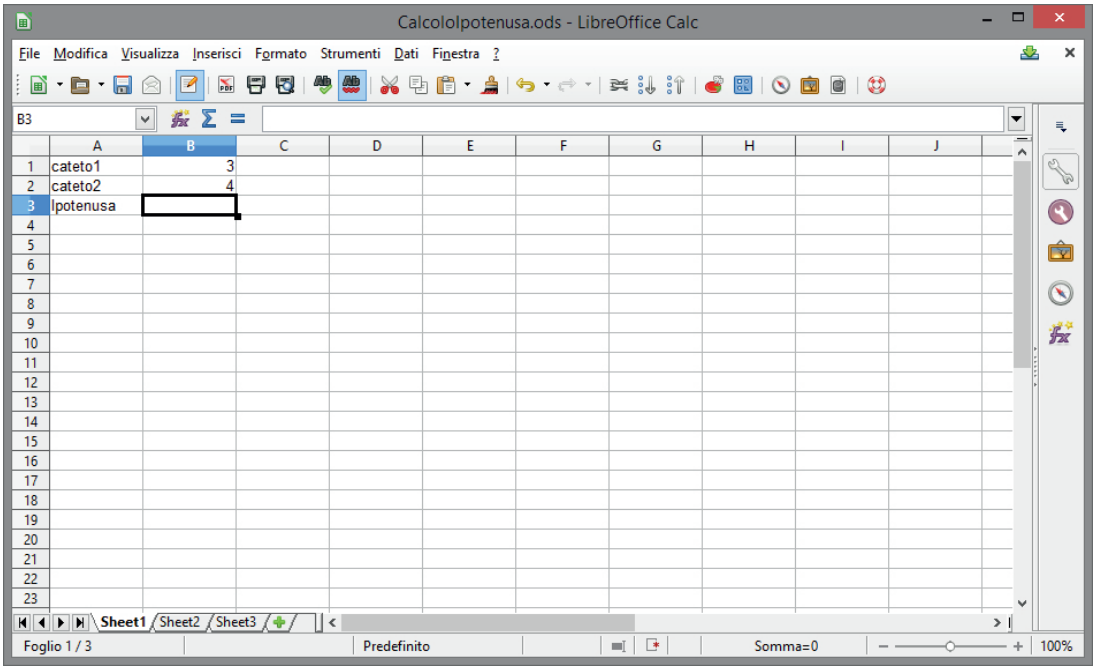
Calcolare l'ipotenusa di un triangolo rettangolo, noti i cateti.

- *Dati di input*: le misure dei due cateti, c1, c2
- *Dati di output*: la misura dell'ipotenusa ipot.
- *Risoluzione*: dopo aver acquisito i valori di *c1* e *c2*, si calcola l'ipotenusa *ipot* con la formula

ipot = $\sqrt{c1^2 + c2^2}$

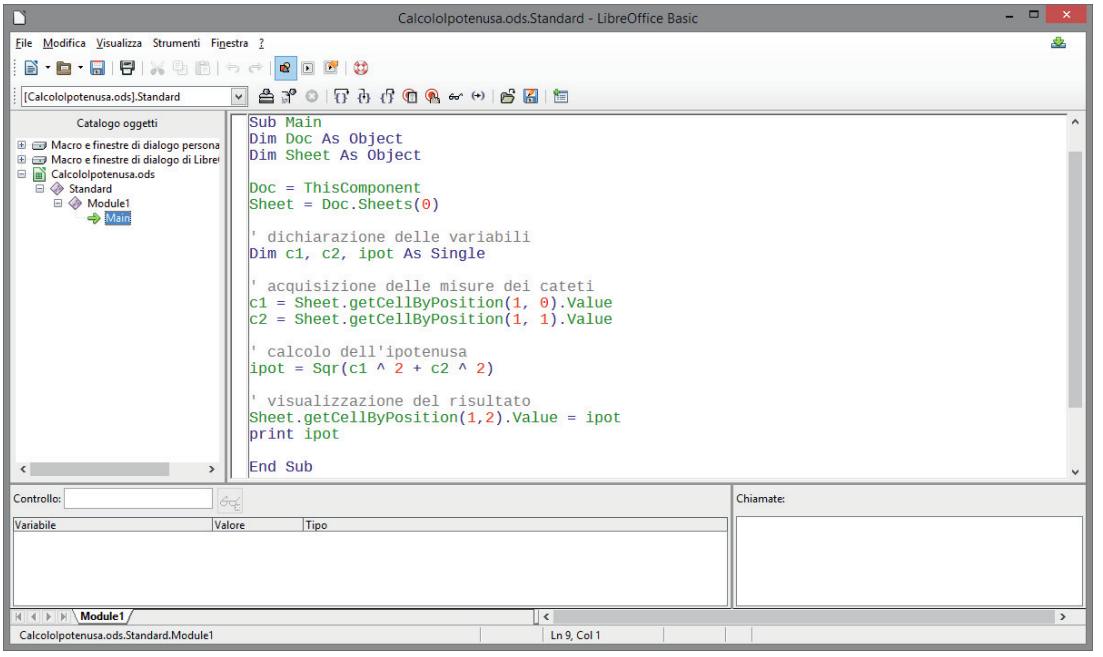
algoritmo TriangoloRettangolo	**riga di intestazione**
variabili	**sezione dichiarativa**
dichiara c1, c2,	
ipot come numeri reali	
inizio	**sezione esecutiva**
immetti c1	
immetti c2	
assegna ipot = $\sqrt{c1^2 + c2^2}$	
scrivi ipot	
fine	

Definiamo la disposizione dei dati sul foglio elettronico *CalcoloIpotenusa.ods*.



Codice

Apriamo la finestra delle macro, come visto in precedenza.
È opportuno visualizzare la barra delle macro con i pulsanti per l'ambiente di sviluppo: menu **Visualizza, Barre degli strumenti**, **Macro**.
Inseriamo il codice per il sottoprogramma *Main*.



Vediamo il significato delle istruzioni precedenti:
Dim (dimensione) elenca i nomi delle variabili utilizzate e specifica il tipo con la clausola **As**; il tipo **Single** rappresenta i numeri reali in singola precisione.
Il tipo **Object** rappresenta genericamente un oggetto utilizzato dal programma: file di Calc, foglio di lavoro, cella del foglio di lavoro.
Gli **operatori aritmetici** sono **+**, **–**, *****, **/** per le quattro operazioni fondamentali (addizione, sottrazione, moltiplicazione e divisione), **^** per l'elevamento a potenza,**** per il quoziente della divisione intera, **MOD** per il resto della divisione intera.
Sqr è la funzione predefinita del linguaggio che calcola la radice quadrata dell'espressione indicata tra le parentesi tonde.
L'**assegnazione** dei valori alle variabili o ad un intervallo (indicata con il segno **=**) è sempre verso sinistra.

Le frasi che iniziano con l'apice sono **frasi di commento**, che servono a documentare le istruzioni del programma.
Nel testo del programma le frasi sono scritte con colori diversi per le parole chiave del linguaggio, le variabili definite dall'utente e le frasi di commento.
ThisComponent rappresenta il file dell'intero foglio di calcolo ed è assegnato alla variabile *Doc* di tipo *Object*:

```
Doc = ThisComponent
```

I fogli di lavoro del file Calc sono organizzati nell'insieme **Sheets** e identificati con un indice che parte dal valore 0. Quindi il primo foglio di lavoro è *Sheets(0)* ed è assegnato alla variabile *Sheet* di tipo *Object*:

```
Sheet = Doc.Sheets(0)
```

Le singole celle del foglio sono identificate attraverso la posizione nella griglia, indicando, nell'ordine, il numero di colonna e il numero di riga, con valori che partono da 0. Quindi la cella B1 si trova nella posizione (1, 0), colonna 2 e riga 1, la cella B2 nella posizione (1, 1), colonna 1 e riga 1, e così via.
La posizione è ottenuta dalla funzione **getCellByPosition**:

```
c1 = Sheet.getCellByPosition(1, 0).Value
c2 = Sheet.getCellByPosition(1, 1).Value
```

La proprietà **Value** consente di acquisire il valore numerico contenuto nella cella, che è un oggetto.
La stessa funzione è poi utilizzata per assegnare alla cella B3 il valore dell'ipotenusa:

```
Sheet.getCellByPosition(1,2).Value = ipot
```

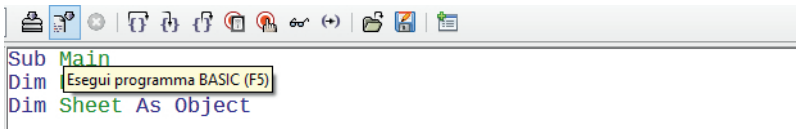
L'istruzione **print** visualizza il valore di una variabile o un messaggio all'interno di una finestra di dialogo:

```
print ipot
```

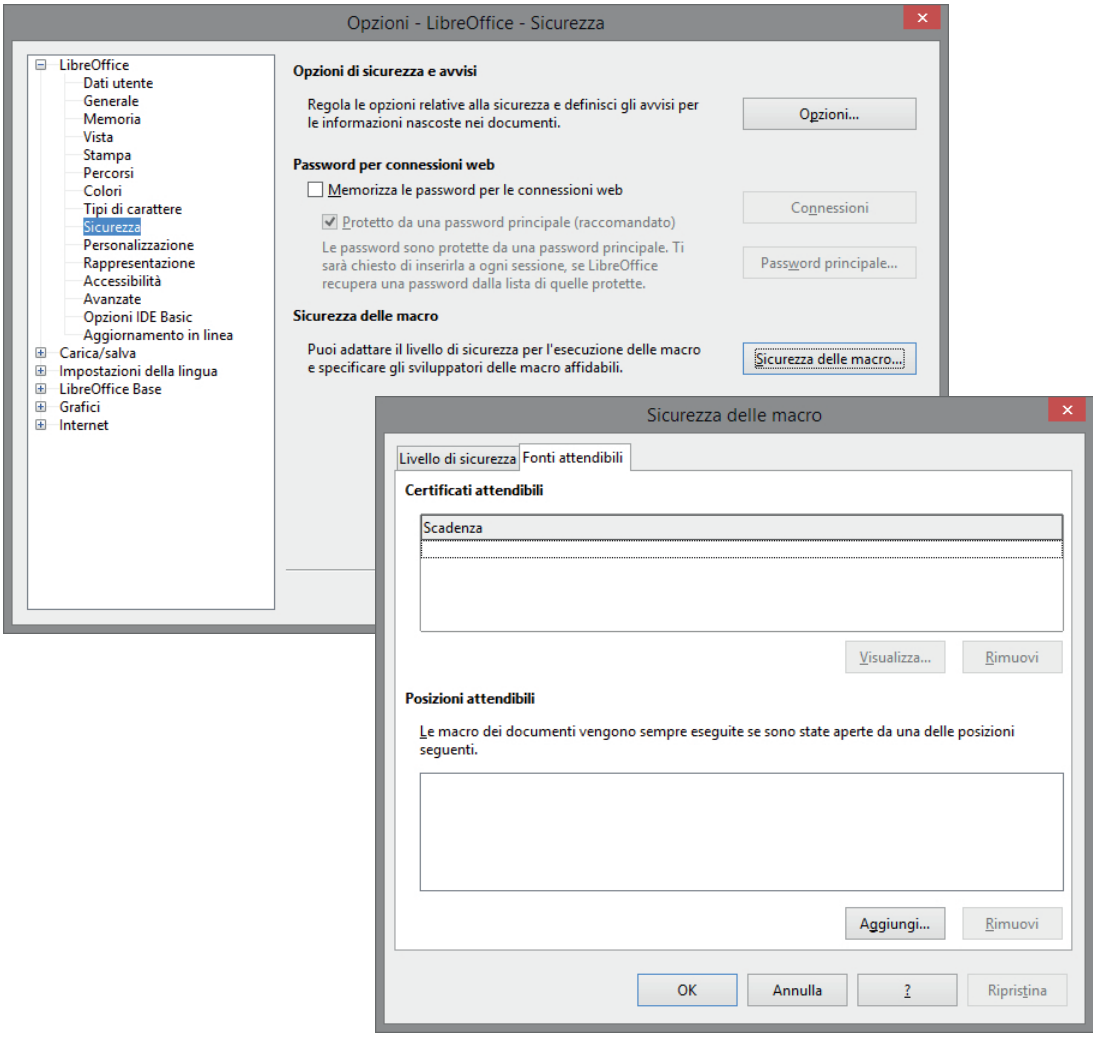
In alternativa si può usare la funzione **MsgBox**:

```
MsgBox ipot
```

Proviamo ora il funzionamento del sottoprogramma.
Nella **Barra Macro**: clic sul pulsante **Esegui programma Basic** (scorciatoia da tastiera: **F5**).



Il sottoprogramma viene eseguito e il valore dell'ipotenusa viene scritto nella cella B3 e visualizzato in una finestra di dialogo.
L'esecuzione delle macro è sottoposta a restrizioni per motivi di sicurezza. Per abilitare l'esecuzione dei sottoprogrammi occorre dichiarare attendibile la provenienza delle macro che si intendono eseguire.
Operando con il proprio computer personale, si possono definire le impostazioni di sicurezza dal menu **Strumenti**, **Opzioni**, **Sicurezza**.
Nella sezione **Sicurezza delle macro**: clic sul pulsante **Sicurezza delle macro** e, nella finestra che si apre, clic sulla scheda **Fonti attendibili**. Attraverso il pulsante **Aggiungi**, inserire il percorso della cartella che contiene il file di Calc.



DICHIARAZIONE DI COSTANTI E VARIABILI

Abbiamo visto nell'esempio precedente che la dichiarazione delle variabili è indicata con la parola chiave **Dim**. Se si scrivono solo i nomi delle variabili, senza specificare il tipo di dato che esse rappresentano (numero intero o reale, oppure stringa), il tipo di dato viene determinato implicitamente dal programma sulla base del dato che viene letto nelle celle del foglio elettronico. Tuttavia, per una maggiore chiarezza nella programmazione e una rappresentazione più precisa del codice, è importante specificare, oltre al nome della variabile, anche il tipo con la clausola **As** (come).

Per esempio:

```
Dim V1 As Integer
```

per dichiarare una variabile V1 di tipo numerico intero.

I tipi di dato più usati sono:
Integer, per variabili memorizzate come numeri interi a 2 byte nell'intervallo da –32.768 a 32.767.
Long, per interi di 4 byte, cioè numeri interi compresi tra –2.147.483.648 e 2.147.483.647.
Single, per numeri reali a virgola mobile e precisione singola a 32 bit (2 byte), compresi tra –3,402823E38 e –1,401298E–45 per valori negativi e tra 1,401298E–45 e 3,402823E38 per valori positivi.
Double, per numeri a virgola mobile e doppia precisione a 64 bit (8 byte) compresi tra –1,79769313486232E308 e –4,94065645841247E–324 per i valori negativi, tra 4,94065645841247E–324 e 1,79769313486232E308 per i valori positivi.
Date, per memorizzare date e orari come numeri reali.
String, per dati composti da una sequenza di caratteri
Boolean, per dati con solo due valori possibili, *True* (–1) o *False* (0).

È possibile dichiarare diverse variabili in una singola istruzione:

```
Dim Contatore As Integer, AreaCerchio As Double, Nome As String
Dim Trovato As Boolean
```

Le costanti sono dichiarate con la parola chiave **Const** in questo modo

```
Const Max = 30
```

Si può, con più precisione, specificare anche il tipo di dato:

```
Const Max As Integer = 30
```

Anche se il codice funziona correttamente senza dichiarare le variabili in modo esplicito con l'istruzione *Dim*, per programmare in modo ordinato è comunque opportuno e utile dichiarare le variabili in modo esplicito e con un tipo di dati specifico. La dichiarazione esplicita consente infatti di ridurre gli errori per conflitti di nome e gli errori di ortografia.
Per evitare che le variabili vengano dichiarate in modo implicito, è possibile posizionare l'istruzione **Option Explicit** prima dei sottoprogrammi nella finestra del codice. Questa istruzione forza la dichiarazione esplicita di tutte le variabili nelle subroutine. Quando viene utilizzata l'istruzione *Option Explicit* e viene incontrato un nome di variabile non dichiarata o digitato in modo non corretto, verrà generato un errore di compilazione (*Variabile non definita*) quando si esegue il programma.

La struttura di selezione

La struttura di **selezione** viene rappresentata secondo lo schema:

IF condizione **THEN**
    istruzione1
ELSE
    istruzione2
END IF

Se la condizione è vera, viene eseguita l'*istruzione1*, altrimenti viene eseguita l'*istruzione2*.
Istruzione1 e *istruzione2* possono indicare, come accade nella maggior parte dei casi, non una sola istruzione, ma un gruppo di istruzioni.
La condizione è un'espressione booleana di cui viene valutata la verità: vengono quindi utilizzati i segni del confronto: <, >, =, >=, <=, <> (*diverso*), e gli operatori booleani AND, NOT, OR per costruire espressioni logiche combinando tra loro più condizioni.

Le strutture di ripetizione per falso e per vero

La struttura di ripetizione può essere rappresentata in modi diversi. La prima forma riguarda la **ripetizione per falso** o **ripetizione post-condizionale**, cioè la ripetizione nella quale il controllo viene fatto in coda, dopo aver eseguito le istruzioni del ciclo.
Questa struttura è tradotta con le istruzioni **DO ... LOOP UNTIL**.

La sintassi generale è la seguente:

DO
    istruzioni
LOOP UNTIL condizione

La ripetizione termina quando la condizione scritta vicino a *Until* diventa vera.
La condizione può contenere anche gli operatori booleani AND, OR, NOT.

La **ripetizione per vero** o **ripetizione con controllo in testa**, è rappresentata con la struttura **DO WHILE ... LOOP**.

La sintassi generale della struttura è la seguente:

DO WHILE condizione
    istruzioni
LOOP

La ripetizione viene eseguita mentre la condizione scritta vicino a *While* si mantiene vera. All'inizio del programma alla variabile *numero* è assegnato il valore 0 per poter forzare l'esecuzione della ripetizione almeno una volta, effettuando il primo controllo della condizione alla partenza della ripetizione.

La struttura di ripetizione con contatore

La **ripetizione con contatore** o **iterazione enumerativa** si rappresenta con la struttura **FOR ... NEXT**.

La sintassi generale della struttura è la seguente:

FOR contatore = iniziale **TO** finale
    istruzioni
NEXT

Le istruzioni racchiuse tra *For* e *Next* sono ripetute tante volte quante occorrono per portare il contatore dal valore iniziale al valore finale incrementandolo di 1 ad ogni ripetizione.