

# Tipi di Dati in Visual Basic

Le variabili di Visual Basic consentono di memorizzare temporaneamente valori durante l'esecuzione di un'applicazione. **Alle variabili è associato un nome, utilizzato per fare riferimento al valore della variabile, e un tipo di dati che determina il tipo di informazioni memorizzate nella variabile.**

Le variabili possono essere considerate segnaposti in memoria per valori sconosciuti.

Per dichiarare una variabile, è necessario utilizzare l'istruzione Dim specificando il nome che si desidera assegnare:

## Dim nomevariabile [As tipo]

Il valore di una variabile in una routine è locale rispetto alla routine e ciò significa che non è possibile accedere a una variabile contenuta in una routine da un'altra routine. In questo modo è possibile utilizzare gli stessi nomi di variabile in routine diverse senza creare conflitti o modificare la routine inavvertitamente.

I nomi di variabile:

- Devono iniziare con una lettera.
- Non possono contenere punti o caratteri per la dichiarazione dei tipi.
- Non possono essere composti da più di 255 caratteri.
- Devono essere univoci all'interno dello stesso form, della stessa routine e area di validità, ovvero l'intervallo valido per i riferimenti alla variabile.

## Area di validità delle variabili

L'area di validità di una variabile definisce le parti del codice in cui la variabile viene riconosciuta. Le variabili dichiarate in una routine possono essere utilizzate o modificate solo dal codice incluso nella routine. In questo caso l'area di validità della variabile è locale rispetto alla routine. In alcuni casi può essere necessario utilizzare variabili con area di validità più ampia in modo che il valore corrispondente sia disponibile, ad esempio, in tutte le routine di un modulo o dell'intera applicazione. In Visual Basic è possibile specificare l'area di validità di una variabile al momento della dichiarazione.

### Definizione dell'area di validità delle variabili

A seconda della modalità di dichiarazione, l'area di validità di una variabile viene definita a livello di routine (locale) o a livello di modulo.

Area di validità	Private	Pubbliche
A livello di routine	Le variabili sono private rispetto alla routine in cui sono inserite.	Non è possibile dichiarare variabili pubbliche in una routine.

A livello di modulo	Le variabili sono private rispetto al modulo in cui sono inserite.	Le variabili sono disponibili in tutti i moduli.
---------------------	--------------------------------------------------------------------	--------------------------------------------------

## Variabili a livello di routine

Le variabili a livello di routine sono riconosciute solo nella routine in cui vengono dichiarate. Tali variabili sono definite anche variabili locali e vengono dichiarate utilizzando le parole chiave Dim o Static. Ad esempio:

**Dim Numero As Integer**

**Static intPermanent As Integer**

I valori delle variabili locali dichiarate con Static esistono per l'intera durata dell'esecuzione dell'applicazione mentre le variabili dichiarate con Dim esistono solo durante l'esecuzione della routine corrispondente.

È consigliabile utilizzare variabili locali per qualsiasi tipo di calcolo temporaneo. È possibile, ad esempio, creare più routine contenenti una variabile **Numero**. Ciascuna routine riconosce la propria versione della variabile a condizione che questa sia dichiarata come variabile locale. Qualsiasi routine può modificare il valore della propria variabile **Numero** senza influire sulle variabili **Numero** delle altre routine.

## Variabili a livello di modulo

Per impostazione predefinita, le variabili a livello di modulo sono disponibili per tutte le routine di un modulo, ma per il codice di altri moduli. Queste variabili devono essere dichiarate con la parola chiave Private nella sezione Dichiarazioni del modulo. Ad esempio:

**Private Numero As Integer**

A livello di modulo non esiste alcuna differenza tra Private e Dim. È tuttavia consigliabile utilizzare Private in quanto si contrappone in modo evidente a Public e agevola pertanto la comprensione del codice.

## Variabili disponibili in tutti i moduli

Per rendere una variabile a livello di modulo disponibile in altri moduli, è necessario dichiararla specificando la parola chiave Public. I valori delle variabili pubbliche sono disponibili in tutte le routine dell'applicazione. In modo analogo alle variabili a livello di modulo, le variabili pubbliche vengono dichiarate nella sezione Dichiarazioni del modulo. Ad esempio:

**Public Numero As Integer**

## Tipi di dati

Il tipo di dati di una variabile determina la modalità di memorizzazione dei bit che rappresentano i valori nella memoria del computer.

Se non si specifica alcun tipo di dati, per impostazione predefinita viene assegnato il tipo di dati **Variant** che consente di rappresentare tipi di dati diversi in più situazioni.

### Tipi di dati numerici

In Visual Basic sono disponibili vari tipi di dati numerici, ovvero **Integer**, **Long** (intero

lungo), **Single** (virgola mobile a precisione semplice), **Double** (virgola mobile a precisione doppia) e **Currency**.

Quando si è certi che in una variabile verranno sempre memorizzati numeri interi, quale il numero 12, anziché numeri frazionari, quale 3,57, è consigliabile dichiararla con il tipo di dati Integer o Long. Le operazioni con valori interi vengono infatti eseguite più rapidamente. Questi tipi di dati utilizzano inoltre una minor quantità di memoria rispetto ad altri tipi e risultano particolarmente utili come variabili contatore in cicli For...Next.

Le variabili che includono valori frazionari devono essere dichiarate con tipo Single, Double o Currency. Il tipo di dati Currency supporta fino a quattro cifre a destra del separatore decimale e fino a quindici cifre a sinistra. Si tratta di un tipo di dati a virgola fissa adatto a calcoli di valuta. Con numeri in virgola mobile (tipi di dati Single e Double) sono disponibili intervalli di valori più ampi rispetto al tipo Currency. Tali valori sono tuttavia soggetti a piccoli errori di arrotondamento.

Il tipo di dati Currency è utile per i calcoli monetari e per calcoli a virgola fissa in cui la precisione riveste un'importanza particolare.

## Il tipo di dati String

Le variabili in cui vengono sempre memorizzate stringhe e mai valori numerici possono essere dichiarate come tipo **String**:

### Private Parola As String

È quindi possibile assegnarvi stringhe e gestirle tramite funzioni stringa:

**Parola = "Database"** oppure **Parola = Left(Parola, 8)**

Per impostazione predefinita, una variabile o argomento di tipo String è una **stringa di lunghezza variabile**, ovvero le dimensioni aumentano o diminuiscono in base ai dati assegnati. È inoltre possibile dichiarare una variabile come **stringa di lunghezza fissa** utilizzando la seguente sintassi:

### Stringa \* dimensioni

Per dichiarare una stringa composta sempre da 50 caratteri, ad esempio, è necessario utilizzare il codice simile al seguente:

### Dim Nominativo As String \* 50

Se si assegna una stringa composta da un numero di caratteri minore di 50, in **Nominativo** vengono automaticamente aggiunti spazi di riempimento finali fino a raggiungere i 50 caratteri. Se invece si assegna una stringa composta da un numero di caratteri maggiore della lunghezza fissa massima prevista, la stringa viene troncata.

Gli spazi di riempimento finali inseriti in stringhe di lunghezza fissa possono essere eliminati con le funzioni **Trim** e **RTrim**.

Le stringhe di lunghezza fissa incluse in moduli standard possono essere dichiarate come Public o Private, mentre in form e in moduli di classe devono essere dichiarate come Private.

## Il tipo di dati Boolean

Le variabili che includono informazioni di tipo vero/falso, sì/no e on/off possono essere dichiarate come tipo Boolean. Il valore predefinito di **Boolean** è False. Nell'esempio seguente,

**Continua** è una variabile Boolean in cui è memorizzata un'impostazione di tipo sì/no.

```
Dim Continua As Boolean
If Valore = 100 Then
    Continua = True
End if
```

## Il tipo di dati Date

I valori di data e ora possono essere memorizzati in variabili sia con tipo di dati specifico **Date** che di tipo Variant. Entrambi i tipi di dati presentano le stesse caratteristiche generali.

Nella conversione di tipi di dati numerici nel tipo Date, i valori a sinistra della parte decimale rappresentano la data, mentre i valori a destra della parte decimale rappresentano l'ora. La mezzanotte corrisponde a 0, il mezzogiorno a 0,5. I numeri interi negativi rappresentano date precedenti al 30 dicembre 1899.

## Riepilogo sui tipi di dati

Tipo di dati	Spazio su disco	Intervallo
<b>Byte</b>	1 byte	Da 0 a 255
<b>Boolean</b>	2 byte	True o False
<b>Integer</b>	2 byte	Da -32.768 a 32.767
<b>Long</b> (intero lungo)	4 byte	Da -2.147.483.648 a 2.147.483.6477
<b>Single</b> (virgola mobile a precisione semplice)	4 byte	Da -3,402823E38 a -1,401298E-45 per valori negativi; da 1,401298E-45 a 3,402823E38 per valori positivi
<b>Double</b> (virgola mobile a precisione doppia)	8 byte	Da -1,79769313486232E308 a -4,94065645841247E-324 per valori negativi; da 4,94065645841247E-324 a 1,79769313486232E308 per valori positivi.
<b>Currency</b> (intero diviso)	8 byte	Da -922.337.203.685.477,5808 a 922.337.203.685.477,5807
<b>Decimal</b>	14 byte	+/-79.228.162.514.264.337.593.543.950.335 senza virgola;



Nella dichiarazione di una matrice, il nome della matrice deve essere seguito dal limite superiore racchiuso tra parentesi:

```
Dim Voti(14) As Integer ' 15 elementi.
```

La prima dichiarazione consente di creare una matrice contenente 15 elementi, con numeri di indice compresi tra 0 e 14.

Il limite inferiore deve essere impostato in modo esplicito con tipo di dati Long specificando la parola chiave **To**:

```
Dim Voti (1 To 15) As Integer
```

Nella dichiarazione precedente, i numeri di indice di **Voti** sono compresi tra 1 e 15.

## Matrici a più dimensioni

In Visual Basic è possibile dichiarare matrici a più dimensioni. La seguente istruzione, ad esempio, dichiara una matrice bidimensionale (10 per 10) in una routine:

```
Static Matrice(9, 9) As Double
```

È possibile dichiarare una delle due dimensioni o entrambe con limiti inferiori espliciti:

```
Static Matrice(1 To 10, 1 To 10) As Double
```

È quindi possibile estendere la matrice a più di due dimensioni:

```
Dim MultiD(3, 1 To 10, 1 To 15)
```

Questa dichiarazione crea una matrice a tre dimensioni: 4 per 10 per 15. Il numero totale di elementi corrisponde al prodotto delle dimensioni, in questo caso 600.

## Matrici dinamiche

In alcuni casi può risultare utile ridimensionare una matrice in fase di esecuzione.

Le matrici dinamiche possono essere ridimensionate in qualsiasi momento. Questo tipo di matrice consente di gestire la memoria in modo efficiente; è possibile, ad esempio, utilizzare una matrice di grandi dimensioni per un periodo di tempo limitato e quindi rendere disponibile la memoria quando non è più necessario utilizzare la matrice.

In alternativa è possibile dichiarare la matrice specificando le dimensioni massime e ignorare quindi gli elementi non necessari. Se tuttavia si applica questo metodo in modo eccessivo, la memoria del sistema operativo potrebbe esaurirsi.

Per dichiarare una matrice dinamica, occorre assegnarle un elenco di dimensioni vuoto:

```
Dim Vettore()
```

Assegnare il numero di elementi desiderato con un'istruzione **ReDim**.

```
ReDim Vettore(X + 1)
```

L'istruzione **ReDim** può essere inserita solo in una routine ed è un'istruzione eseguibile, ovvero causa l'esecuzione di un'operazione dell'applicazione in fase di esecuzione.

La sintassi dell'istruzione **ReDim** è la stessa utilizzata per matrici fisse. Per ogni dimensione di una matrice è possibile utilizzare **ReDim** per modificare sia il numero di elementi che i limiti inferiore e superiore. Non è tuttavia possibile modificare il numero di dimensioni.

## Costanti

Il codice include spesso valori costanti ripetuti più volte oppure dipende da determinati valori numerici difficili da ricordare in quanto non hanno alcun significato evidente.

In questi casi è possibile agevolare la lettura e la manutenzione del codice tramite l'utilizzo di costanti. Una costante è un nome significativo che rappresenta più valori numerici o stringhe non soggetti a modifiche. A differenza delle variabili, non è possibile modificare una costante o assegnarvi un nuovo nome.

La sintassi per la dichiarazione di costanti è la seguente:

**[Public|Private] Const nomecostante[As tipo] = espressione**

L'istruzione **Const** può rappresentare un valore matematico o di data/ora; istruzione può inoltre essere utilizzata per la definizione di costanti stringa:

**Public Const Posti As Integer = 9                      Const DataInizio = #1/1/95#**

**Const Password = "Enigma"**

È inoltre possibile definire costanti in base a costanti definite precedentemente:

**Const Doppio = PiGreco \* 2**

## Variabili statiche

Oltre all'area di validità, alle variabili è associato un *ciclo di vita*, ovvero il periodo di tempo durante il quale conservano il proprio valore. I valori di variabili pubbliche e a livello di modulo vengono conservati per l'intera durata del ciclo di vita dell'applicazione. Le variabili locali dichiarate con Dim esistono tuttavia solo durante l'esecuzione della routine in cui sono dichiarate. In genere, dopo l'esecuzione di una routine, i valori delle corrispondenti variabili locali non vengono conservati e la memoria da queste utilizzata viene recuperata. Alla successiva esecuzione della routine, tutte le corrispondenti variabili locali vengono reinizializzate.

È tuttavia possibile conservare il valore delle variabili locali rendendo le variabile *statiche*. Per eseguire questa operazione, è necessario dichiarare una o più variabili in una routine specificando la parola chiave **Static**, esattamente come con l'istruzione Dim:

### Static Depth

La seguente funzione, ad esempio, consente di calcolare un totale progressivo aggiungendo un nuovo valore al totale dei precedenti valori memorizzati nella variabile statica **ApplesSold**:

```
Function RunningTotal(num)
    Static ApplesSold
    ApplesSold = ApplesSold + num
    RunningTotal = ApplesSold
End Function
```

Se **ApplesSold** viene dichiarata con Dim anziché con Static, i valori accumulati precedenti non vengono conservati in tutte le chiamate alla funzione, che pertanto restituirà lo stesso valore con cui è stata chiamata.

È possibile ottenere lo stesso risultato dichiarando **ApplesSold** nella sezione Dichiarazioni del modulo in modo da renderla una variabile a livello di modulo. Dopo aver modificato l'area di validità in questo modo, tuttavia, la routine non ha più accesso esclusivo alla variabile. Dato che altre routine possono accedere e modificare il valore della variabile, i totali risultano poco affidabili e la manutenzione del codice risulta più difficile.

(Fonte Guida il linea di MSDN Library)