SEQUENZE DI CARATTERI E TIPO STRING

Il Tipo di Dato string

Una Variabile di **Tipo string** permette di memorizzare una **Stringa** ossia una **Sequenza di Caratteri**.

☞ Se vuoi dichiarare una *variabile di Tipo string* e darle il *nome St* allora devi scrivere: **string St**

I Caratteri di una Stringa vengono memorizzati secondo la Codifica **UniCode**, quindi si ha una *occupazione di memoria di 2 byte* per ogni carattere memorizzato.

Una singola Variabile di Tipo STRING può contenere fino a un massimo di circa *2 miliardi di caratteri*.

I **Valori di Tipo String**, ossia le *Sequenze di Caratteri* che devono essere memorizzate nelle variabili String, nel codice C#, devono sempre essere *delimitate fra una* **Coppia di Doppi Apici**.

☞ Se vuoi assegnare alla *variabile stringa St*, la sequenza di caratteri *ciao mamma*, devi scrivere: **St = "ciao mamma"**

Il valore **Stringa Vuota** è una stringa costituita da 0 caratteri: si indica *digitando due "doppi apici" in successione* (**" "**)

☞ Spesso la *stringa vuota* si usa per *inizializzare* una variabile stringa, scrivendo: **string St = " "**

Concatenamento di Stringhe e Operatore +

In Visual C#, quando l'operatore "**+**" è utilizzato fra due stringhe, è chiamato **Operatore di Concatenamento** (**operatore +**) e consente di creare una nuova stringa, *concatenando* (ossia *unendo*) due stringhe esistenti.

☞ L'istruzione seguente: **St = "Prof." + "Rossi"** memorizza in St la sequenza **"Prof.Rossi"**

Le **Espressioni String** sono vere e proprie espressioni (calcoli) che, *anziché avere argomenti numerici, elaborano e hanno come risultato delle Stringhe,* ossia Sequenze di Caratteri. Esse possono quindi contenere *Variabili String*, *Operatori* (come il Concatenamento +), *Valori String* (da scrivere fra doppi apici) e altre funzioni.

☞ Se in una *variabile string Nome* è memorizzato il valore "**Mario**", allora l'istruzione seguente:

St = "Il mio migliore amico è " + Nome

memorizza nella *variabile string St* la sequenza di caratteri **"Il mio migliore amico è Mario".**

Come vedi, nell'*Espressione String* a destra dell'uguale, sono presenti *Variabili String* (**Nome**), *Operatori* (**+**) e *Valori String* (**"Il mio migliore amico è "**).

Nelle *Espressioni String* è possibile anche **inserire variabili o espressioni numeriche**, perché esse vengono *automaticamente convertite in sequenze di caratteri* e trattate quindi come stringhe.

✋ In una *variabile intera Pos*, è memorizzata la *posizione di arrivo* di un pilota e in una *variabile string NomePilota* è memorizzato il *nome del pilota* stesso. Considera la seguente istruzione di assegnazione:

St = "Il pilota " + NomePilota + " si è piazzato in " + Pos + "° posizione."

Se *NomePilota* contiene "**Charles Leclerc**" e *Pos* contiene *1* allora in St troviamo il valore:

"Il pilota Charles Leclerc si è piazzato in 1° posizione."

Il valore della variabile Pos (ossia 1), pur essendo di tipo *int*, viene automaticamente convertito nel *valore string "1"* e regolarmente concatenato alle altre stringhe. Nota la presenza degli SPAZI nelle sequenze per distaccare le parole.

Lunghezza di una Stringa e Proprietà LENGTH

La **Proprietà Length** di una Stringa restituisce la **Lunghezza della Stringa**, ossia un *numero intero* che indica il *Numero di Caratteri memorizzati nella Stringa*.

<*stringa*>**.Length** *… restituisce il Numero di Caratteri presenti nella stringa, quindi un valore di tipo* ***intero*** *…*

✋ Non dimenticare che, come tutte le proprietà, *Length* "*restituisce*" un valore, quindi va utilizzata in una *espressione*.

☞ Se nella *variabile stringa St* è memorizzato il valore "*ciao mamma*", allora scrivendo, ad esempio:

X = St.Length

memorizzi nella *variabile int X* il valore 10, ossia il *numero di caratteri* che compongono "*ciao mamma*".
Attenzione, perché lo *spazio* (**blank**), per C#, è un carattere come gli altri.

Costruzione di Stringhe

Spesso è necessario **"Costruire" delle Stringhe**, aggiungendovi un carattere (o una sequenza di caratteri) per volta.

☞ Ad esempio, creare una stringa composta da tanti caratteri uguali (es. "**#######**") o elencare, in un'unca stringa, tutti i dati di una lista di numeri, magari separati da una virgola (es. "**5, 6, 2, 7, 12, 4**"), ecc.

L'operazione (simile alla *somma di una lista di numeri*) si realizza utilizzando *un ciclo* che, ad ogni passo, "*aggiunge*" alla fine della stringa, un altro dato. L'istruzione da ripetere è del tipo **St = St +** …

☞ Ad esempio, con il seguente ciclo crei una stringa composta dai primi **10** numeri separati da spazio:

```
string St = " ";                 … dichiara la stringa da creare e la inizializza a stringa vuota (0 caratteri)
for ( int K = 1;  K <= 10;  K++) … il ciclo ripete per 10 volte con K che varia da 1 a 10
      St = St +  K + " ";        … aggiunge, alla fine della stringa St, il valore di K e uno spazio ( " " )
```

Per **Aggiungere alla Fine di una Stringa** (St) si usa l'istruzione: **St = St +** <*espressione-stringa*>
Per **Aggiungere all'Inizio di una Stringa** (St) si usa l'istruzione: **St =** <*espressione-stringa*> **+ St**

Ordinamento e Confronto di Stringhe

In *C#* è possibile effettuare un **Confronto fra Due Stringhe** allo scopo di determinare quale, fra le due stringhe, è "minore" (ossia "viene prima") *secondo l'ordine alfabetico*.

✋ Per confrontare le sequenze di caratteri presenti in due stringhe, è possibile usare gli operatori "**==**" (*uguale*) e "**!=**" (*diverso*) al fine di stabilirne l'uguaglianza, ma **non è possibile usare gli operatori di confronto ">", "<", ">=", "<="**

Il *Confronto* che utilizzeremo (chiamato **Confronto Ordinale**) si effettua con il metodo:

string.CompareOrdinal (<*stringa1*>, <*stringa2*>) ⇨ intero

Il metodo *CompareOrdinal* **restituisce un numero intero** il cui valore può essere:

minore di zero	… allora <*stringa1*> ***è minore di*** <*stringa2*>
uguale a zero	… allora <*stringa1*> ***è uguale*** *a* <*stringa2*>
maggiore di zero	… allora <*stringa1*> ***è maggiore di*** <*stringa2*>

☞ Se vuoi usare questa funzione in una *istruzione if*, per confrontare due Nomi chiesti in input, puoi scrivere:

```
string PrimoNome = txtNome1.Text;
string SecondoNome = txtNome2.Text;
if ( string.CompareOrdinal ( PrimoNome, SecondoNome ) < 0 )
        MessageBox.Show ("Viene prima " + PrimoNome);
else
        MessageBox.Show ("Viene prima " + SecondoNome);
```

Nota Bene: il *Confronto Ordinale* si basa sull'ordinamento definito dal **Codice ASCII**, per cui i risultati possono a volte essere diversi da quanto ci si aspetterebbe dall'ordinamento naturale delle parole.

✋ Ad esempio, la stringa "**Z**ippo" risulterà minore della stringa "**p**ippo" perché le lettere minuscole hanno tutte Codice ASCII superiore a quelle maiuscole (**maiuscole da 65 a 90** e **minuscole da 97 a 122**).