Con questo articolo vedremo come si trattano gli array di caratteri, concludendo così il capitolo sui vettori. Per parlarne, mi rifarò al manuale di Scarpa piuttosto che a quello di Nelson, dal momento che spiega considerando utilizzata la libreria *Io* che possiede già alcune funzioni preformate che semplificano di gran lunga la vita (e, di fatto, questo articolo fondamentalmente mostra proprio queste funzioni).

```
Include "Parser";

Include "Verblib";

Include "Replace";


Array stringa buffer "Ciao Mondo!";


[ Funzione;

PrintStringArray(stringa);

PressKey();

];

[ Initialise;

funzione();

quit;

];

Include "Io";

Include "ItalianG";
```

Questo semplice listato ci fa vedere come si dichiara un array di tipo stringa: vediamo sempre che lo definiamo come Array, dopodiché gli assegniamo il nome (in questo caso proprio "stringa") e poi, al posto della solita "freccina", scriviamo la parola *buffer* seguita dal testo che vogliamo contenga, incluso tra doppie virgolette. Come potete vedere, il testo non è necessariamente un'unica parola.
[Nota: a differenza degli array che abbiamo visto le altre volte, questo utilizza le prime due caselle (di indice 0 e 1) per immagazzinare certe informazioni, quindi la prima lettera della stringa, "C", è contenuta nella casella di indice 2.
Un'altra importante differenza tra gli array visti negli esempi precedenti e gli array stringhe è che se inserissimo una stringa costituita da cifre, questa non verrebbe interpretata come un numero di tot cifre, ma come un insieme di caratteri corrispondenti ai numeri: per questo motivo non vi azzardate a fare operazioni aritmetiche su vettori stringhe, nemmeno se rappresentano dei numeri, il risultato sarebbe disastroso - vedere esercizio]
Ogni volta che vorremo stampare a schermo il contenuto di questo array ci basterà usare la funzione **PrintStringArray()** nelle cui parentesi va inserito il nome dell'array da stampare.

Vediamo inoltre nella routine funzione l'istruzione **PressKey()**: questa dice al programma di aspettare che il giocatore prema un tasto qualsiasi per continuare l'esecuzione.
Questo è il caso base, ma, come dice Scarpa nel suo manuale, questi tipi di array vengono utilizzati per "contenere" più stringhe (diverse nel corso dell'avventura). Vediamo come:

```
Include "Parser";

Include "VerbLib";

Include "Replace";

Constant LUNGH_MAX 23;

Array stringa->LUNGH_MAX;

[ Funzione;

PrintToBuffer(stringa, LUNGH_MAX, "Prima stringa");

PrintStringArray(stringa);

print "^";

PrintToBuffer(stringa, LUNGH_MAX, "Seconda stringa");

PrintStringArray(stringa);

PressKey();

];

[ Initialise;

Funzione();

quit;

];

Include "Io";

Include "ItalianG";
```

Qual è la differenza rispetto al primo listato? Innanzitutto vediamo che abbiamo dichiarato l'array nel modo "classico".
L'istruzione **PrintToBuffer(**a,b,c**)** serve per dire al programma: copia nell'array a di lunghezza massima b la stringa c. Vediamo che questo è un procedimento distruttivo, nel senso che la stringa precedentemente inserita viene cancellata definitivamente.
[Nota: è necessario che la lunghezza massima ecceda sempre di 3 il numero di caratteri che la stringa può contenere, senza superare il valore di 160: nel nostro esempio, l'array stringa può contenere al massimo 20 caratteri]

Perché questo è un procedimento importante? Perché permette, ad esempio, di registrare negli array anche ciò che digita il giocatore. Ecco come:

```
Include "Parser";

Include "VerbLib";

Include "Replace";


Constant LUNGH_MAX 18;

Array nome_giocatore->LUNGH_MAX;

Array nome_programmatore buffer "Mariano";


[ Funzione;

print "Digita il tuo nome:^^> ";

ReadArray(nome_giocatore, LUNGH_MAX);

nome_giocatore->2 = UpperCase(nome_giocatore->2); !###

print "Salve, ", (PrintStringArray)nome_giocatore , ", io mi chiamo " ,
(PrintStringArray)nome_programmatore, ".";

if((CmpStr(nome_giocatore, nome_programmatore))==1) print "^Siamo omonimi!^";

PressKey();

];


[ Initialise;

Funzione();

quit;];


Include "Io";

Include "ItalianG";
```

Vediamo che abbiamo l'array nome_programmatore già definito, contenente la stringa "Mariano", mentre teniamo "libero" l'array nome_giocatore. Guardando la routine *Funzione* vediamo che è chiesto al giocatore di digitare il suo nome. L'istruzione **ReadArray(**a,b**)** funziona un po' come la *PrintToBuffer*, con l'unica differenza che legge i caratteri dalla tastiera (ossia quelli digitati dal giocatore) e li memorizza nell'array a di lunghezza massima b. Questa funzione è molto utile, ma ha un inghippo: rende tutti i caratteri minuscoli. Per questo è stata creata l'istruzione **UpperCase()** che

permette di rendere maiuscolo il carattere che vogliamo (come visto precedentemente, il primo carattere di una stringa occupa la posizione 2, quindi l'istruzione contenuta nella riga commentata con ### significa, leggendola da destra a sinistra: accedi alla cella di indice 2 e maiuscolizza il carattere, poi metti il carattere ottenuto nella cella di indice 2 dell'array nome_giocatore).
Vediamo inoltre l'istruzione **CmprStr(**a,b**)** che permette di comparare 2 stringhe a, b e che restituisce il valore 1 se queste sono identiche.
Naturalmente sulle stringhe, essendo comunque array, possono essere eseguite le operazioni già viste per gli array (ma non le operazioni aritmetiche!): prendiamo ad esempio questo codice:

```
Include "Parser";

Include "VerbLib";

Include "Replace";


Constant LUNGH_MAX 18;

Array nome_giocatore->LUNGH_MAX;


[ Funzione i;

print "Come ti chiami?^^> ";

ReadArray(nome_giocatore, LUNGH_MAX);

for (i=2: i<nome_giocatore->1: i++)

if (nome_giocatore->i=='a') nome_giocatore->i='u';

nome_giocatore->2 = UpperCase(nome_giocatore->2);

print "Hai detto che ti chiami ", (PrintStringArray)nome_giocatore , "?^Credo di non aver capito bene!";

PressKey();

];


[ Initialise;

Funzione();

quit;];


Include "Io";

Include "ItalianG";
```

Se non capite cosa fa, provate a compilarlo e a giocare (se il vostro nome non contiene la lettera "a" siete liberissimi di usare il mio nome!).
Da notare, ancora una volta, che questo ciclo *for* comincia dal valore 2, perché è lì che si trova la prima lettera della stringa! E, ancora, notare che la condizione di termine ciclo non è stata espressa da *i < LUNGH_MAX*, ma da *i < nome_giocatore->1*, questo proprio perché la lunghezza massima di caratteri presente nell'array è memorizzata nell'array stesso (nella casella di indice 1)!

Per finire lascio un esercizio: cercate di riprodurre il comportamento di questo programma. Per semplificare, fingiamo che ogni volta che il giocatore lo apre agisca ingenuamente e che digiti sempre la password che gli è indicata sul foglietto.
Cercate anche di produrre un messaggio di morte nel caso si sbagli password anche la seconda volta.