Versione 1.0

ATTENZIONE – COPIA SERIALIZZATA

Il pacchetto xConsole® è protetto dalle leggi sul diritto d'autore ed il suo utilizzo è soggetto all'approvazione, da parte del cliente, del contratto di licenza che specifica chiaramente l'ambito di utilizzo. **In nessun caso il pacchetto deve essere distribuito o trasmesso a terzi in violazione dei limiti imposti dal contratto di licenza. Poiché ogni copia del prodotto è serializzata, la SZP è sempre in grado a partire da una copia abusiva di risalire al licenziatario originale, contro il quale saranno esercitate tutte le azioni legali previste per la violazione della legge sul diritto d'autore e del contratto di licenza.**

SOMMARIO

Introduzione

xConsole® è un controllo Active-X che permette di realizzare velocemente ed efficacemente applicativi a carattere utilizzando linguaggi di programmazione come Visual Basic, Delphi, Visual C++ ed altri.

Gli applicativi a carattere che si ottengono attraverso xConsole® sono caratterizzati da performance a 32 bit e consentono di sfruttare tutte le potenzialità dell'ambiente Windows e dei linguaggi di programmazione che accolgono il controllo.

xConsole® nasce dall'esigenza di avere uno strumento semplice e flessibile, che consenta di gestire facilmente l'ingresso di testo, la selezione di opzioni, menu, ecc.

La realizzazione di applicativi a carattere è un'esigenza sempre presente, anche nei sistemi operativi grafici – spesso la semplicità di una console è preferita per scrivere utilità di sistema da eseguire sulla riga di comando; inoltre, l'interfaccia a carattere è indispensabile per creare applicativi da eseguire attraverso Telnet Server (la maggior parte dei terminali in radiofrequenza incorporano dei client a carattere in emulazione VT o Ansi).

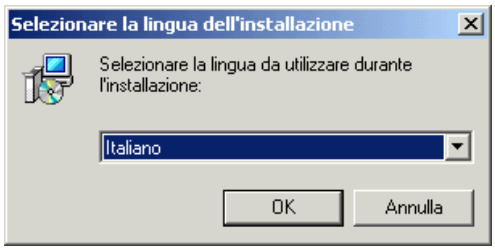
I vantaggi offerti da xConsole® sono molteplici:

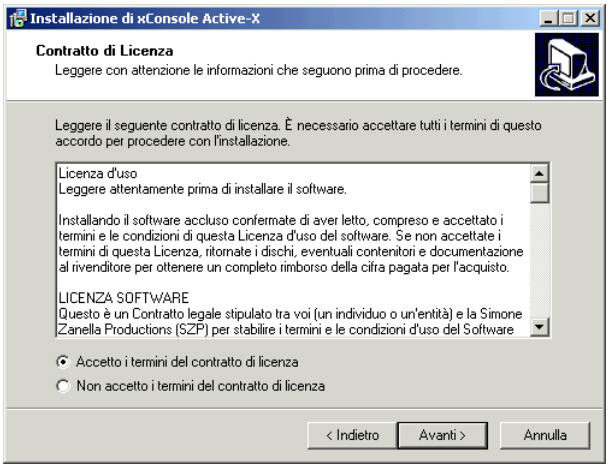
1. permette di ottenere applicativi con interfaccia a carattere anche da linguaggi che non supportano tale modalità (es. Visual Basic) oppure hanno funzionalità molto rudimentali (es. Delphi);
2. grazie ad un sistema di eventi, permette di personalizzare completamente le varie fasi dell'ingresso dati;
3. rende disponibili funzioni di lettura stringhe con scrolling e caratteristiche avanzate di controllo (maschere, espressioni regolari semplici ed estese, validazione di date, ore, numeri interi e decimali, campi password, ecc.);
4. offre liste di opzioni a scelta singola o multipla, menu, campi di testo multiriga, disegno di linee, cornici, ombre, salvataggio e ripristino dello schermo, stampa di messaggi accompagnati o meno da pulsanti di conferma, ecc.

Sebbene le API di Windows permettano di creare delle console a carattere e gestire la stampa a video e l'ingresso di tasti, le funzioni altamente ottimizzate e collaudate offerte da xConsole® costituiscono un risparmio di tempo enorme nella scrittura di questo tipo di programmi e semplificano notevolmente il codice delle proprie applicazioni.

Installazione

Per installare il pacchetto, inserire il cd-rom nel lettore del proprio PC, equipaggiato con sistema operativo Windows 95/98/ME/NT/2000/XP o superiore. Nel caso in cui non parta in automatico il programma di installazione, aprire da “Gestione Risorse” il disco corrispondente al cd-rom ed avviare il programma SETUP.EXE.





Selezionare la lingua per il pacchetto da installare e premere Ok; leggere il contratto di licenza, selezionare “Accetto i termini del contratto di licenza” e premere Avanti. Alle schermate successive premere sempre Avanti ed attendere il completamento dell’installazione. Al termine, premere “Fine”.
Il componente Active-X sarà così registrato e potrà essere utilizzato nel proprio ambiente di sviluppo.

Utilizzo del controllo xConsole® in Visual Basic

Il controllo xConsole® è un componente Active-X; in quanto tale, per utilizzarlo in un proprio applicativo è necessario indicare all'ambiente di sviluppo che si intende farne uso.

In Visual Basic, ciò si ottiene aggiungendolo nella lista dei componenti (voce del menu "Progetto"); a questo punto, comparirà nella casella degli strumenti una piccola icona che rappresenta una X rossa sopra una C azzurra. Il controllo può quindi essere selezionato e posizionato sul form, come qualsiasi altro controllo di Visual Basic. Poiché l'applicazione sviluppata utilizzerà un'interfaccia a carattere, ci sarà un unico form, con attributo *Visible* impostato a *False*.

Il proprio codice dovrà essere scritto in un modulo, facendo riferimento al componente caricato nel form; infatti, poiché l'interfaccia a carattere non dipende da eventi ma segue un flusso più o meno lineare, verrebbe a mancare un punto di ingresso per l'avvio dell'elaborazione (l'utilizzo dell'evento *Form_load* è sconsigliato).

Il primo metodo da richiamare è InitConsole (*True*), che alloca la console necessaria all'invocazione dei successivi metodi ed imposta i valori di default delle proprietà.

Se l'applicazione che si sta sviluppando è una utility da invocare a linea di comando potrebbe essere utile salvare le dimensioni ed il contenuto dello schermo attuale (utilizzando le proprietà MaxCol, MaxRow ed il metodo ScreenSave), per poterlo poi ripristinare subito prima dell'uscita.

Di seguito, l'utente può impostare i colori secondo il proprio gusto personale, oppure in base alle limitazioni del terminale remoto sul quale l'applicazione andrà successivamente in esecuzione; un metodo semplice per determinare se il programma è in esecuzione all'interno dell'IDE di Visual Basic oppure è compilato sono le seguenti istruzioni:

```
Err.Clear
On Error Resume Next
Debug.Print 1 / 0
' Se c'è errore, sono nell'IDE
If Err.Number <> 0 Then
  ' Sono nell'IDE
Else
  ' Applicazione compilata
End If
```

Sempre per contemplare eventuali limitazioni dei terminali remoti, può essere necessario invocare il metodo Resize per ridurre lo schermo della console alle dimensioni di quello del terminale.

A questo punto si possono impostare i valori di lunghezza, tipo giustificazione, cornice, ombra ed iniziare ad invocare i metodi per disegnare l'interfaccia utente della propria applicazione.
Si consiglia di compilare le diverse schermate dell'applicazione con un editor di testo che mostra la posizione (riga, colonna), in modo da avere un riferimento sul quale basare il proprio codice.
Tutti gli eventi del controllo andranno introdotti nel form principale; si possono ottenere facendo doppio click sul controllo XConsole e scegliendo poi l'evento rilevante.

Prima di terminare l'applicazione, si consiglia di invocare il metodo ShutDownConsole per liberare la console allocata, ma solo se non si è all'interno dell'ambiente di sviluppo.

Una volta compilata l'applicazione, è necessario modificarne il tipo; infatti, Visual Basic produce esclusivamente applicazioni grafiche. Per la corretta esecuzione del programma compilato bisogna quindi indicare al sistema operativo che l'eseguibile è un applicativo console (cioè a carattere).

Per far ciò, si possono seguire due strade:
a) utilizzare l'utility a riga di comando EditBin, installata con Visual Studio;
b) utilizzare l'utility a riga di comando ConsoleMode, fornita gratuitamente con il componente Active-X.

La sintassi di invocazione di EditBin è la seguente:

```
editbin /SUBSYSTEM:CONSOLE program.exe
```

Il funzionamento di ConsoleMode è ancora più semplice:

```
consolemode program.exe
```

In entrambi i casi, al termine la vostra applicazione sarà pronta ad essere eseguita da una console o un telnet server.

Si rimanda al codice dell'applicativo di esempio "CodQt" per ulteriori dettagli e suggerimenti.

Utilizzo del controllo xConsole® in Visual C++

Il controllo xConsole® è un componente Active-X; in quanto tale, per utilizzarlo in un proprio applicativo è necessario indicare all'ambiente di sviluppo che si intende farne uso.

In Visual C++, è necessario seguire i seguenti passi per creare un'applicazione con xConsole®:
1. creare un nuovo progetto e selezionare "MFC AppWizard (exe)"; indicare il nome del progetto e premere OK;
2. allo Step 1, indicare "Dialog based" e premere Next;
3. allo Step 2, selezionare solo "ActiveX Controls" e "Automation" (eventualmente specificare "Windows Sockets" se richiesti) e premere Next;
4. allo Step 4, indicare se richiesti i commenti ed il tipo di collegamento della libreria MFC e premere Next;
5. premere Finish per completare la generazione dei file di supporto.

A questo punto, dalla finestra di dialogo del programma selezionare:

Project > Add to Project > Components and Controls

Da "Registered ActiveX Controls", scegliere XCONSOLE Control e premere Insert; confermare con Ok l'inserimento.
Lasciare il nome della classe a CXCONSOLE e modificare i nomi dei file "header" e "implementation" a:

```
XCONSOL1.h
XCONSOL1.cpp
```

Confermare con Ok; al termine dell'inserimento, chiudere la finestra di dialogo per l'introduzione dei componenti.
Nella palette dei controlli apparirà l'icona di XConsole; selezionarla e posizionare il controllo sulla finestra di dialogo.
Impostare la proprietà "Visible" della finestra di dialogo a False (finestra non visibile).

Dal menu, selezionare Edit > ClassWizard; premere il tab Member Variables; impostare come nome della classe quello della finestra di dialogo e come Control_ID quello del nuovo controllo XConsole appena inserito (default: IDC_XCONSOLECTRL1).
Premere Add Variable e impostare come nome della variabile `m_xc`; premere Ok e confermare con Ok la chiusura di ClassWizard.

Aprire il file sorgente (cpp) contenente l'implementazione della finestra di dialogo e cercare la funzione OnInitDialog; all'interno di questa, dopo la riga di commento:

```
// TODO: Add extra initialization here
```

inserire la seguente riga:

```
m_xc.InitConsole(TRUE);
```

seguita dall'invocazione della funzione di ingresso vero e proprio del programma, che andrà a gestire l'interfaccia attraverso XCONSOLE, utilizzando la variabile `m_xc` che sarà passata sotto forma di un puntatore ad un oggetto CXCONSOLE.

Al ritorno della funzione di ingresso, inserire una chiamata a PostMessage che provochi l'uscita dalla finestra di dialogo (che viene utilizzata solo per ospitare il controllo xConsole®):

```
[nome dialogo]::PostMessage(WM_CLOSE, 0, 0);
```

Ovviamente, `[nome dialogo]` sarà sostituita dal nome della propria finestra di dialogo (nel caso dell'esempio, CXcdemoDlg).

RICORDARSI DI ELIMINARE EVENTUALI FLAG DI COMPILAZIONE DOUBLE-BYTE: xConsole® supporta solo caratteri a singolo byte.

Se l'applicazione che si sta sviluppando è una utility da invocare a linea di comando potrebbe essere utile salvare le dimensioni ed il contenuto dello schermo attuale (utilizzando le proprietà MaxCol, MaxRow ed il metodo ScreenSave), per poterlo poi ripristinare subito prima dell'uscita.

Di seguito, l'utente può impostare i colori secondo il proprio gusto personale, oppure in base alle limitazioni del terminale remoto sul quale l'applicazione andrà successivamente in esecuzione.

Sempre per contemplare eventuali limitazioni dei terminali remoti, può essere necessario invocare il metodo Resize per ridurre lo schermo della console alle dimensioni di quello del terminale.

A questo punto si possono impostare i valori di lunghezza, tipo giustificazione, cornice, ombra ed iniziare ad invocare i metodi per disegnare l'interfaccia utente della propria applicazione.
Si consiglia di compilare le diverse schermate dell'applicazione con un editor di testo che mostra la posizione (riga, colonna), in modo da avere un riferimento sul quale basare il proprio codice.

Per gestire gli eventi restituiti dal controllo, è necessario introdurre delle funzioni di gestione, utilizzando la seguente procedura:
1. dal menu View, selezionare ClassWizard;
2. fare click sulla linguetta "Message Maps";
3. selezionare dalla casella "Class name" la finestra di dialogo che contiene il controllo xConsole;
4. nella casella "Object Ids" selezionare l'ID del controllo ActiveX (es. IDX_XCONSOLECTRL1). La finestra "Messages" mostrerà la lista degli eventi di xConsole®; un evento per il quale è già definita una funzione di gestione è mostrato in grassetto;
5. selezionare l'evento che si desidera gestire; premere "Add function" per aggiungere una funzione, oppure "Edit Code" per visualizzare nel sorgente la funzione di gestione e modificarla opportunamente.

Prima di terminare l'applicazione, si consiglia di invocare il metodo ShutDownConsole per liberare la console allocata.

Una volta compilata l'applicazione, è necessario modificarne il tipo; infatti, Visual C++ produrrà un'applicazione grafica. Per la corretta esecuzione del programma compilato bisogna quindi indicare al sistema operativo che l'eseguibile è un applicativo console (cioè a carattere).

Per far ciò, si possono seguire due strade:
a) utilizzare l'utility a riga di comando EditBin, installata con Visual Studio;
b) utilizzare l'utility a riga di comando ConsoleMode, fornita gratuitamente con il componente Active-X.

La sintassi di invocazione di EditBin è la seguente:

```
editbin /SUBSYSTEM:CONSOLE program.exe
```

Il funzionamento di ConsoleMode è ancora più semplice:

```
consolemode program.exe
```

In entrambi i casi, al termine la vostra applicazione sarà pronta ad essere eseguita da una console o un telnet server.

Si rimanda al codice dell'applicativo di esempio "XCDEMO" per ulteriori dettagli e suggerimenti.

Metodi di xConsole®

Di seguito sono descritti tutti i metodi del controllo xConsole®, incluse le loro interazioni (sottolineate attraverso un prefisso comune).

I metodi sono indicati in **BLU**, le proprietà in **ROSSO**, gli eventi in **VERDE**.

Le costanti sono indicate sempre attraverso identificatori mnemonici; nel modulo XCONSOLE.BAS e nel file header XCONSOLE.H compaiono i corrispondenti valori effettivi.

La sintassi indicata segue quella di Visual Basic (prima riga, in blu); i metodi corrispondenti in Visual C++ sono indicati in grigio (seconda riga); inoltre, valgono le seguenti conversioni di tipo:

Tipo Visual Basic	Tipo Visual C++
Boolean	BOOL
Integer	short o short * (se passato per riferimento)
Long	long o long * (se passato per riferimento)
String	LPCTSTR (parametro nei metodi) CString (valore di proprietà) BSTR (valore ritornato da un metodo)

Nota bene:

1) Tutti i metodi che hanno per parametri (impliciti o espliciti) le coordinate X ed Y vi addizionano sempre il valore delle proprietà OffsetX e OffsetY, le quali permettono quindi di spostare rapidamente in qualsiasi parte dello schermo la propria maschera, senza modificare manualmente alcuna coordinata.

2) Per ragioni di efficienza, il controllo xConsole® limita al minimo indispensabile la verifica sui parametri con i quali sono invocati i metodi; si ponga particolare attenzione, in particolare, a specificare sempre coordinate all'interno dell'area dello schermo effettivamente allocata.

Lista alfabetica dei metodi

AboutBox ()
void AboutBox()

Apre a video una finestra di dialogo grafica con informazioni sulla versione del controllo e sul copyright. Non ritorna alcun valore. E' l'unico metodo che produce un'uscita grafica.

Alert (*ByVal Tag as Long*) *as Boolean*
BOOL Alert(long Tag);

Apre a video un box contenente un testo e dei pulsanti. Ritorna True se l'utente ha selezionato un pulsante, False se è uscito con Esc. Un solo pulsante alla volta è visualizzato a video. Il comportamento del metodo è influenzato dalle seguenti proprietà:

AlertText *as String*	Messaggio da stampare nel box
AlertButtons *as String*	Testo da visualizzare nei pulsanti; la stringa deve avere il seguente formato: "tasto 1[#tasto 2..]", cioè bisogna

	usare il carattere “#” per separare i vari pulsanti
AlertCurrentButton *as Integer*	Numero del pulsante predefinito (se invalido, il primo pulsante diviene quello predefinito); questa proprietà è aggiornata al termine della selezione, anche se l’utente preme Esc
AlertBackColor *as Integer* AlertForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa del testo del messaggio
AlertButtonBackColor *as Integer* AlertButtonForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa dei pulsanti
AlertFrameForeColor *as Integer*	Colore di primo piano della cornice (i pulsanti hanno come colore della cornice AlertButtonForeColor)
Frame *as Boolean* Frame3D *as Boolean* FrameChars *as String* ShadowMode *as Integer*	Stato della cornice, tipo e caratteri utilizzati per disegnarla; tipo di ombra (assente, destra, sinistra)

Il parametro *Tag* determina la reazione del metodo all’introduzione di dati da parte dell’operatore; se è zero, il comportamento di default è il seguente:

tasti cursore	Cambiano il pulsante visualizzato, permettendo di selezionare la risposta desiderata
Enter, spazio	Accettano la selezione corrente
Esc	Annulla

Se *Tag* è diverso da zero, ad ogni tasto premuto viene generato il seguente evento:

AlertKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)

dove:

KeyAscii *as Integer*	Contiene il tasto introdotto dall’utente; può essere aggiornato per simulare la pressione di un diverso tasto e impostato a 0 per scartarlo
Action *as String*	Determina l’azione richiesta in risposta dal metodo; può essere aggiornato con uno dei valori indicati nella tabella più sotto
Tag *as Integer*	Numero identificativo dell’Alert

Il valore di *Tag* permette di conoscere qual è l’Alert attivo, potendo in questo modo adottare comportamenti diversi a seconda delle necessità.

I valori possibili per *Action* sono i seguenti:

ALERT_ACCEPT	Elabora normalmente il tasto
ALERT_DISCARD	Ignora il tasto (corrisponde ad impostare *KeyAscii* a 0 indicando ALERT_ACCEPT come *Action*)
ALERT_SELECT	Seleziona il pulsante AlertCurrentButton e rimuove il box
ALERT_SELECTNR	Seleziona il pulsante AlertCurrentButton e ritorna lasciando il box (no restore)
ALERT_ABORT	Annulla e ritorna rimuovendo il box
ALERT_ABORTNR	Annulla e ritorna lasciando il box (no restore)
ALERT_NEXT	Prossimo pulsante

ALERT_PREVIOUS	Pulsante precedente
ALERT_FIRST	Primo pulsante
ALERT_LAST	Ultimo pulsante

Attribute () *as Integer*
short Attribute();
AttributeXY (*ByVal X as Integer, ByVal Y as Integer*) *as Integer*
short AttributeXY(short X, short Y);

Ritorna l'attributo video alle coordinate correnti o specificate. L'attributo combina i colori di primo piano e sfondo; è possibile ottenere i due colori componenti applicando il metodo AttributeSplit.

AttributeJoin (*ByVal ForegroundColor as Integer, ByVal BackgroundColor as Integer*) *as Integer*
short AttributeJoin(short ForegroundColor, short BackgroundColor);

Ritorna l'attributo video corrispondente alla combinazione dei colori di sfondo e primo piano specificati; è l'inverso del metodo AttributeSplit.

AttributeSplit (*ByVal Color as Integer, ByRef ForegroundColor as Integer, ByRef BackgroundColor as Integer*)
void AttributeSplit(short Color, short* ForegroundColor, short* BackgroundColor);

Suddivide l'attributo video *Color* nei corrispondenti colori di primo piano (*ForegroundColor*) e sfondo (*BackgroundColor*); è l'inverso del metodo AttributeJoin.

Box (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer, ByVal Bottom as Integer*)
void Box(short Left, short Top, short Right, short Bottom);

Disegna una cornice tra le coordinate (*Left, Top*) e (*Right, Bottom*); le caratteristiche della cornice tracciata dipendono dalle seguenti proprietà:

Frame3D *as Boolean*	Indica se la cornice debba avere un aspetto a rilievo (due lati adiacenti hanno una colorazione più scura rispetto agli altri due lati)
FrameChars *as String*	Stringa di 8 bytes che rappresentano in senso orario i caratteri da utilizzare per disegnare la cornice, partendo dall'angolo superiore sinistro ed andando in senso orario. Il default, dopo InitConsole, permette di tracciare cornici a tratto singolo (FRAME_SINGLE) sfruttando i caratteri semigrafici. Le costanti, FRAME_DOUBLE, FRAME_SNGDOU, FRAME_DOUSNG e FRAME_DOTS possono essere utilizzate in alternativa.
FrameBackColor *as Integer* FrameForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa della cornice
Pattern *as Integer*	Codice ASCII del carattere utilizzato per cancellare l'interno della cornice (default: 32, corrispondente a spazio).
ShadowMode *as Integer*	Tipo di ombra, per dare evidenza alla cornice; i possibili valori sono: NO_SHADOW = nessuna ombra SHADOW_LEFT = ombra a sinistra

	SHADOW_RIGHT = ombra a destra

ClearArea (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer, ByVal Bottom as Integer, ByVal Color as Integer, ByVal Pattern as Integer*)
void ClearArea(short Left, short Top, short Right, short Bottom, short Color, short Pattern);

Cancella l'area compresa tra (*Left, Top*) e (*Right, Bottom*), utilizzando l'attributo *Color* ed il carattere ASCII corrispondente a *Pattern*.

Cls ()
void Cls();

Cancella tutto lo schermo, utilizzando il carattere di riempimento ed i colori di primo piano/sfondo correnti.

ColorizeArea (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer, ByVal Bottom as Integer, ByVal ForeColor as Integer, ByVal BackColor as Integer*)
void ColorizeArea(short Left, short Top, short Right, short Bottom, short ForeColor, short BackColor);

Sostituisce i colori di primo piano e sfondo sull'area specificata, rimpiazzandoli con quelli indicati.

GetMaxColRow ()
void GetMaxColRow();

Carica nelle proprietà MaxCol e MaxRow rispettivamente il numero di colonne ed il numero di righe dello schermo. Normalmente, non è necessario invocare questo metodo, in quanto InitConsole e Resize aggiornano automaticamente il valore delle due proprietà.

GetXY ()
void GetXY();

Carica nelle proprietà X e Y le coordinate correnti del cursore.

GotoXY (*ByVal X as Integer, ByVal Y as Integer*)
void GotoXY(short X, short Y);

Sposta il cursore alle coordinate indicate ed aggiorna le proprietà X e Y; è possibile ottenere lo stesso risultato assegnando dei valori alle medesime proprietà.

HPrint (*ByVal Text as String*) *as Integer*
short HPrint(LPCTSTR Text);
HPrintXY (*ByVal X as Integer, ByVal Y as Integer, ByVal Text as String*) *as Integer*
short HPrintXY(short X, short Y, LPCTSTR Text);

Stampa alle coordinate correnti (oppure a quelle specificate) la stringa *Text*, evidenziando ogni carattere preceduto dal simbolo "~"; ritorna il numero di righe utilizzate (oppure –1 se la giustificazione è impossibile). Le caratteristiche del testo stampato dipendono dalle seguenti proprietà:

ForegroundColor *as Integer* BackgroundColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa

Justification *as Integer*	Giustificazione del testo; può assumere i seguenti valori: J_NOJUST = Nessuna giustificazione J_LEFT = Giustificazione a sinistra J_CENTER = Al centro J_RIGHT = Giustificazione a destra J_JUST = Giustificazione a pacchetto
JustificationLength *as Integer*	Lunghezza nella quale effettuare la giustificazione (dovrebbe essere sempre maggiore o uguale alla lunghezza del testo da giustificare)
Pattern *as Integer*	Codice ASCII del carattere utilizzato per la giustificazione

Se *X* = 0, la funzione entra in modo emulazione, nel quale nulla viene stampato a video ma sono calcolate e restituite solo le righe necessarie alla stampa.

HiColor (*ByVal Color as Integer*) *as Integer*
short HiColor(short Color);

Ritorna l'attributo corrispondente all'evidenziazione di *Color*. Questa è la funzione di trasformazione utilizzata per l'evidenziazione dei caratteri fatta da HPrint ed altre funzioni.

InitConsole (*ByVal UseExisting as Boolean*)
void InitConsole(long UseExisting);

Inizializza il controllo ed imposta a default tutte le proprietà; è necessario invocare sempre questo metodo prima di utilizzare qualsiasi altro metodo o proprietà del controllo. Il valore *UseExisting* specifica se riutilizzare un'eventuale console già associata al processo (True) oppure se crearne una nuova (False); di norma, si dovrebbe sempre specificare True.

InputString (*ByVal Tag as Integer*) *as Boolean*
BOOL InputString(long Tag);
InputStringXY (*ByVal X as Integer, ByVal Y as Integer, ByVal Tag as Integer*) *as Boolean*
BOOL InputStringXY(short X, short Y, long Tag);

Accetta la digitazione di una stringa, eventualmente controllandone il formato; la prima forma utilizza le coordinate correnti, la seconda quelle specificate. Ritorna True se l'utente ha confermato con Enter, False se è uscito con Esc. Il comportamento del metodo è influenzato dalle seguenti proprietà:

InputDefault *as String*	Valore iniziale della stringa (al ritorno contiene il buffer editato); questa proprietà è aggiornata anche se l'utente preme Esc
InputMaxLength *as Integer*	Massima lunghezza accettata per la stringa
InputWindowLength *as Integer*	Lunghezza della finestra di editing
InputStartPos *as Integer*	Posizione nella riga di editing; questa proprietà è aggiornata al termine, anche se l'utente preme Esc
InputWindowOffset *as Integer*	Offset della finestra di editing; questa proprietà è aggiornata al termine, anche se l'utente preme Esc
InsertMode *as Boolean*	Stato di inserimento: se True, ogni carattere digitato viene inserito, spostando in avanti gli eventuali caratteri seguenti; se False, il carattere sovrascrive quanto esistente; questa proprietà è aggiornata al termine, anche

	se l'utente preme Esc
InputBackColor *as Integer* InputForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la finestra di editing
InputPicture *as String*	Stringa per la validazione automatica di quanto digitato; se vuota, non viene effettuato alcun controllo. La tabella sotto illustra i possibili valori.
DateType *as Integer* Epoch *as Integer*	Formato della data ed epoca, utilizzati per la validazione delle date.
SilentMode *as Boolean*	Se True, nessuna segnalazione acustica sarà emessa in caso di mancata validazione (sarà comunque invocato l'evento SoundRequest)

I caratteri di InputPicture hanno il seguente significato:

!	converte tutte le lettere alfabetiche in maiuscolo
*	stampa asterischi al posto dei caratteri introdotti (es. richiesta password)
N	accetta solo un numero intero
F	accetta solo un numero intero o decimale
D	accetta solo una data; l'interpretazione della data dipende dalle proprietà DateType ed Epoch. La prima può assumere i seguenti valori: DATE_US = formato americano (mese/giorno/anno) DATE_EUROPE = formato europeo (giorno/mese/anno) DATE_JAPAN = formato giapponese (anno/mese/giorno) L'epoca determina come devono essere interpretati gli anni nelle date contratte, cioé quelle date in cui per l'anno sono utilizzate solo due cifre; in questo caso, se decine ed unità sono inferiori alle decine ed unità dell'anno di riferimento in Epoch, viene assunto come riferimento il secolo successivo, altrimenti il secolo attuale; esempio: Epoch = 1970 Anno di: 01/01/69 = 2069 Anno di: 01/01/70 = 1970 Anno di: 01/01/97 = 1997
H	accetta solo un'ora nel formato "hh:mm:ssx", dove mm e ss vanno da 0 a 59, hh va da 1 a 12 (se x è "p" o "a") oppure da 0 a 23 (se x è uno spazio o manca); x deve essere 'a', 'A', 'p', 'P' oppure uno spazio
R*s*	accetta solo se soddisfa *s* (espressione regolare estesa)
T*s*	accetta solo se soddisfa *s* (espressione regolare estesa case-unsensitive)
P*s*	accetta solo se soddisfa *s* (espressione regolare)
O*s*	accetta solo se soddisfa *s* (espressione regolare case-unsensitive)
M*s*	maschera per l'introduzione dei dati; *s* può contenere i seguenti caratteri: X = carattere qualunque N = cifra 0-9 O = cifra 0-7 H = cifra 0-9 o A-H B = cifra 0 o 1 A = carattere alfabetico

	U = carattere alfanumerico altro = carattere letterale

Per un approfondimento sulle espressioni regolari si consulti il capitolo ad esse dedicato.

Il parametro *Tag* determina la reazione del metodo all'introduzione di dati da parte dell'operatore; se è zero, il comportamento di default è il seguente:

tasti cursore destra e sinistra	Spostano il cursore nel buffer di editing
Enter	Conferma quanto introdotto; se la stringa non soddisfa le regole di validazione, viene emesso un segnale acustico e l'utente rimane in modifica
Backspace/Canc	Cancella il carattere precedente oppure l'attuale
Esc	Annulla
Ins	Cambia tra modalità inserimento e sovrascrittura (viene modificata la forma del cursore ed il valore di InsertMode)
Home	Porta il cursore ad inizio riga
End	Porta il cursore alla fine della stringa
Altro carattere compreso tra 32 e 255	Introduce il carattere nella stringa (se soddisfa le eventuali regole di validazione)

Se *Tag* è diverso da zero, ad ogni tasto premuto viene generato il seguente evento:

InputKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)

dove:

KeyAscii *as Integer*	contiene il tasto introdotto dall'utente; può essere aggiornato per simulare la pressione di un diverso tasto e impostato a 0 per scartarlo
Action *as String*	determina l'azione richiesta in risposta dal metodo; può essere aggiornato con uno dei valori indicati nella tabella più sotto
Tag *as Integer*	numero identificativo della InputString

Il valore di *Tag* permette di conoscere qual è l'InputString attiva, potendo in questo modo adottare comportamenti diversi a seconda delle necessità.

I valori possibili per *Action* sono i seguenti:

INPUT_ACCEPT	Accetta il carattere ed inseriscilo nella stringa
INPUT_UPDATE	InputDefault modificato; aggiorna il buffer di editing e continua la modifica
INPUT_UPDATEANDCONFIRM	InputDefault modificato; aggiorna il buffer di editing ed accetta la nuova stringa
INPUT_ABORT	Annulla l'input
INPUT_CONFIRM	Conferma l'input
INPUT_DISCARD	Scarta il carattere
INPUT_LEFT	Sposta il cursore a sinistra
INPUT_RIGHT	Sposta il cursore a destra
INPUT_HOME	Sposta il cursore all'inizio

INPUT_END	Sposta il cursore alla fine

KeyHit () *as Long*
long KeyHit();

Ritorna il codice del prossimo tasto nel buffer di tastiera, oppure zero se il buffer è vuoto. Questo metodo ritorna immediatamente; il tasto non viene comunque rimosso dal buffer di tastiera. KeyHit considera anche eventuali tasti introdotti attraverso il metodo KeyStuff. Utilizzare KeyInput o KeyInputTimed per leggere il tasto e rimuoverlo dal buffer.

KeyInput () *as Long*
long KeyInput();

Attende la pressione di un tasto e ne ritorna il codice; considera eventuali tasti introdotti attraverso KeyStuff. Questo metodo arresta l'esecuzione del programma finché un tasto non diviene disponibile; utilizzare KeyHit per determinare la disponibilità di un tasto nel buffer senza interrompere il programma e senza rimuoverlo dal buffer di tastiera. Utilizzare KeyInputTimed se si desidera impostare un timeout per l'attesa di un tasto.

KeyInputTimed (*ByVal Seconds as Integer*) *as Long*
long KeyInputTimed(short Seconds);

Attende la pressione di un tasto per *Seconds* secondi e ne ritorna il codice; ritorna zero in caso di timeout. Tiene conto di eventuali tasti introdotti attraverso KeyStuff. Questo metodo arresta l'esecuzione del programma finché un tasto non diviene disponibile oppure finché non trascorrono i secondi indicati – se *Seconds* è zero diviene funzionalmente equivalente a KeyInput. Utilizzare KeyHit per determinare la disponibilità di un tasto nel buffer senza interrompere il programma e senza rimuoverlo dal buffer di tastiera.

KeyStuff (*ByVal KeyAscii as Long*)
void KeyStuff(long KeyAscii);

Introduce il tasto *KeyAscii* nel buffer di tastiera; tutti i metodi di xConsole® tengono conto dei tasti introdotti con questo metodo, esattamente come se l'utente li avesse introdotti attraverso la tastiera.

LineFromTo (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer, ByVal Bottom as Integer*)
void LineFromTo(short Left, short Top, short Right, short Bottom);

Disegna una linea da (*Left, Top*) a (*Right, Bottom*), utilizzando il carattere 1 della proprietà LineCharsHV se orizzontale o il carattere 2 se verticale; se LineCharsHV non è definita o è incorretta, utilizza i caratteri 2 e 4 della proprietà FrameChars. Utilizza i colori FrameForeColor e FrameBackColor.
Ovviamente, è possibile tracciare solo linee orizzontali o verticali.

List (*ByVal Tag as Integer*) *as Boolean*
BOOL List(long Tag);
ListXY (*ByVal X as Integer, ByVal Y as Integer, ByVal Tag as Integer*) *as Boolean*
BOOL ListXY(short sX, short sY, long Tag);

Apre a video un box contenente una lista di opzioni; la prima forma utilizza le coordinate correnti, la seconda quelle specificate. Ritorna True se l'utente ha confermato la selezione,

False se è uscito con Esc. Il comportamento del metodo è influenzato dalle seguenti proprietà:

ListTitle *as String*	Titolo della lista di opzioni (stampato sulla cornice in alto, al centro); visualizzato solo se la cornice è attiva
ListOptions *as String*	Testo delle opzioni; la stringa deve avere il seguente formato: “opzione 1[#opzione 2..]”, cioè bisogna usare il carattere “#” per separare le varie opzioni
ListCurrentOption *as Integer*	Numero d'ordine dell'opzione predefinita; questa proprietà è aggiornata al termine della selezione, anche se l’utente preme Esc
ListRows *as Integer*	Numero di righe visibili
ListColumns *as Integer*	Larghezza visible (se 0, corrisponde alla larghezza della riga più lunga)
ListWindowOffset *as Integer*	Offset di stampa delle opzioni (numero di caratteri da saltare all’inizio di ciascuna opzione); questa proprietà è aggiornata al termine della selezione, anche se l’utente preme Esc
ListCurrentLine *as Integer*	Linea alla quale viene visualizzata l'opzione selezionata (default: 0); viene aggiornata all'uscita
ListMultiSelect *as Boolean*	Permette o meno la selezione multipla
ListSelection *as Integer*	Codice ASCII del carattere utilizzato per indicare la spunta di un’opzione quando è attiva la selezione multipla (default: 16)
ListMap *as String*	stringa vuota (se ListMultiSelect = False), oppure stringa di “0” e “1” che indica lo stato di selezione delle varie opzioni (“0” = non selezionata, “1” = selezionata); questa proprietà è aggiornata al termine della selezione, anche se l’utente preme Esc
TitleBackColor *as Integer* TitleForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa del titolo
SelectedBackColor *as Integer* SelectedForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa dell’opzione evidenziata
UnselectedBackColor *as Integer* UnselectedForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa delle opzioni non evidenziate e dello sfondo della cornice
ListFrameForeColor *as Integer*	Colore di primo piano della cornice e dell’indicatore di posizione a colonna
Frame *as Boolean* Frame3D *as Boolean* FrameChars *as String* ShadowMode *as Integer*	stato della cornice, tipo e caratteri utilizzati per disegnarla; tipo di ombra (assente, destra, sinistra)

Il parametro *Tag* determina la reazione del metodo all’introduzione di dati da parte dell’operatore; se è zero, il comportamento di default è il seguente:

tasti cursore	Cambiano l’opzione evidenziata (su/giù) oppure l’offset di visualizzazione dell’opzione corrente (destra/sinistra)
Enter	Seleziona l’opzione corrente (ed esce, se ListMultiSelect è False)

Spazio	Seleziona l'opzione corrente e passa alla successiva (solo se ListMultiSelect è True)
Esc	Annulla
Ctrl+Enter	Conferma la selezione (solo se ListMultiSelect è True)
Tab	Cambia lo stato di selezione di tutte le voci (seleziona o deseleziona tutte le opzioni)
Home	Evidenzia la prima opzione
End	Evidenzia l'ultima opzione

Se *Tag* è diverso da zero, ad ogni tasto premuto viene generato il seguente evento:

ListKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)

dove:

KeyAscii *as Integer*	contiene il tasto introdotto dall'utente; può essere aggiornato per simulare la pressione di un diverso tasto e impostato a 0 per scartarlo
Action *as String*	determina l'azione richiesta in risposta dal metodo; può essere aggiornato con uno dei valori indicati nella tabella più sotto
Tag *as Integer*	numero identificativo del List

Il valore di *Tag* permette di conoscere qual è il List attivo, potendo in questo modo adottare comportamenti diversi a seconda delle necessità.

I valori possibili per *Action* sono i seguenti:

LIST_ACCEPT	Elabora il tasto
LIST_DISCARD	Ignora il tasto premuto
LIST_SELECT	Seleziona l'opzione ListCurrentOption (e termina rimuovendo il box, se non multipla)
LIST_SELECTNR	Seleziona l'opzione ListCurrentOption (e termina lasciando il box, se non multipla)
LIST_ENDSEL	Solo per multipla: conferma selezione e rimuove il box della lista
LIST_ENDSELNR	Solo per multipla: conferma selezione e lascia il box della lista (no restore)
LIST_ABORT	Abbandona e rimuove il box della lista
LIST_ABORTNR	Abbandona e lascia il box della lista (no restore)
LIST_ENHANCE	Evidenzia ListCurrentOption, che diventa la nuova opzione corrente
LIST_NEXT	Passa alla successiva opzione
LIST_PREVIOUS	Passa alla precedente opzione
LIST_FIRST	Passa alla prima opzione
LIST_LAST	Passa all'ultima opzione
LIST_REDRAW	Ridisegna la lista (ListWindowOffset modificato)

Menu (*ByVal Tag as Integer*) *as Boolean*
BOOL Menu(long Tag);
MenuXY (*ByVal X as Integer, ByVal Y as Integer, ByVal Tag as Integer*) *as Boolean*
BOOL MenuXY(short X, short Y, long Tag);

Apre a video un box contenente un menu di opzioni; la prima forma utilizza le coordinate correnti, la seconda quelle specificate. Ritorna True se l'utente ha scelto una voce, False se è uscito con Esc. Il comportamento del metodo è influenzato dalle seguenti proprietà:

MenuOptions *as String*	Opzioni del menu; la stringa deve avere il seguente formato: "opzione 1[|descrizione 1]#opzione 2[|descrizione 2].." cioè bisogna usare il carattere "#" per separare le varie voci ed il carattere "|" per separare la voce di menu dalla sua descrizione. Se testo*x* è vuoto, descrizione*x* è il carattere con cui disegnare la riga di separazione; lasciare solo "|" per ottenere una riga vuota.
MenuUnselectable *as String*	stringa di "0" (opzione selezionabile) o "1" (opzione non selezionabile); se l'opzione non ha carattere corrispondente in MenuUnselectable allora è selezionabile
MenuCurrentOption *as Integer*	Numero d'ordine dell'opzione predefinita; questa proprietà è aggiornata al termine della selezione, anche se l'utente preme Esc
SelectedBackColor *as Integer* SelectedForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa dell'opzione evidenziata
UnselectedBackColor *as Integer* UnselectedForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa delle opzioni non evidenziate e dello sfondo della cornice
UnselectableBackColor *as Integer* UnselectableForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa delle opzioni non selezionabili
MenuFrameForeColor *as Integer*	Colore di primo piano della cornice
Frame *as Boolean* Frame3D *as Boolean* FrameChars *as String* ShadowMode *as Integer*	stato della cornice, tipo e caratteri utilizzati per disegnarla; tipo di ombra (assente, destra, sinistra)
ScoreboardBackColor *as Integer* ScoreboardForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa delle descrizioni
ScoreboardStatus *as Boolean* ScoreboardX *as Integer* ScoreboardY *as Integer* ScoreboardJustification *as Integer* ScoreboardLength *as Integer*	Stato di abilitazione per la visualizzazione, posizione di stampa, tipo giustificazione e lunghezza riga per le descrizioni

Le descrizioni sono una breve nota informativa che accompagna ciascuna voce di menu; esse vengono visualizzate non appena si evidenzia un'opzione, con le modalità specificate attraverso le proprietà Scoreboard[..].

Il parametro *Tag* determina la reazione del metodo all'introduzione di dati da parte dell'operatore; se è zero, il comportamento di default è il seguente:

Freccia su	Evidenzia l'opzione precedente
Freccia giù	Evidenzia l'opzione successiva
Enter, freccia destra, Spazio	Seleziona l'opzione corrente

Esc	Annulla
Home, PagSu	Evidenzia la prima opzione
End, PagGiù	Evidenzia l'ultima opzione

Il carattere preceduto da "~" (ASCII 126) all'interno del testo di un'opzione appare evidenziato nella stampa a video e costituisce il tasto per la selezione diretta della voce di menu.

Se *Tag* è diverso da zero, ad ogni tasto premuto viene generato il seguente evento:

MenuKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)

dove:

KeyAscii *as Integer*	contiene il tasto introdotto dall'utente; può essere aggiornato per simulare la pressione di un diverso tasto e impostato a 0 per scartarlo
Action *as String*	determina l'azione richiesta in risposta dal metodo; può essere aggiornato con uno dei valori indicati nella tabella più sotto
Tag *as Integer*	numero identificativo del Menu

Il valore di *Tag* permette di conoscere qual è il Menu attivo, potendo in questo modo adottare comportamenti diversi a seconda delle necessità.

I valori possibili per *Action* sono i seguenti:

MENU_ACCEPT	elabora il tasto
MENU_DISCARD	ignora il tasto premuto
MENU_SELECT	seleziona MenuCurrentOption, ritorna True
MENU_SELECTNR	seleziona MenuCurrentOption, ritorna True e lascia il box del menu (no restore)
MENU_ABORT	ritorna False e rimuove il box del menu
MENU_ABORTNR	ritorna False e lascia il box del menu
MENU_ENHANCE	evidenzia MenuCurrentOption, che diventa la nuova opzione corrente
MENU_NEXT	passa alla successiva opzione
MENU_PREVIOUS	passa alla precedente opzione
MENU_FIRST	passa alla prima opzione
MENU_LAST	passa all'ultima opzione
MENU_ENABLE	rivaluta MenuUnselectable, cambiando lo stato di selezionabilità delle varie voci

OSD (*ByVal Text as String*) *as String*

CString OSD(LPCTSTR Text);

Apre una finestra a centro schermo con il testo specificato e ritorna una stringa utilizzabile dal metodo OSDRestore per ripristinare il contenuto del video sottostante. Il comportamento del metodo è influenzato dalle seguenti proprietà:

AlertBackColor *as Integer* AlertForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa del testo del messaggio
AlertFrameForeColor *as Integer*	Colore di primo piano della cornice

Frame *as Boolean* Frame3D *as Boolean* FrameChars *as String* ShadowMode *as Integer*	Stato della cornice, tipo e caratteri utilizzati per disegnarla; tipo di ombra (assente, destra, sinistra)

Eventuali caratteri preceduti da "~" sono stampati evidenziati.

OSDRestore (*ByVal Screen as String*)
void OSDRestore(LPCTSTR OSDBuffer);

Ripristina il contenuto dello schermo sottostante un box OSD; il contenuto di *Screen* deve essere stato ottenuto mediante una precedente chiamata del metodo OSD.

Resize (*ByVal Width as Integer, ByVal Height as Integer*)
void Resize(short Width, short Height);

Cambia le dimensioni della console a quelle indicate (se possibile); in caso di successo, le proprietà MaxCol e MaxRow divengono rispettivamente uguali a *Width* e *Height*.

ReverseArea (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer, ByVal Bottom as Integer*)
void ReverseArea(short Left, short Top, short Right, short Bottom);

Inverte i colori sull'area del video compresa tra (*Left, Top*) e (*Right, Bottom*).

SPrint (*ByVal Text as String*) *as Integer*
short SPrint(LPCTSTR Text);
SPrintXY (*ByVal X as Integer, ByVal Y as Integer, ByVal Text as String*) *as Integer*
short SPrintXY(short X, short Y, LPCTSTR Text);

Stampa alle coordinate correnti (oppure a quelle specificate) la stringa *Text*; ritorna il numero di righe utilizzate (oppure –1 se la giustificazione è impossibile). Le caratteristiche del testo stampato dipendono dalle seguenti proprietà:

ForegroundColor *as Integer* BackgroundColor *as Integer*	Colori di sfondo e primo piano utilizzati per la stampa
Justification *as Integer*	Giustificazione del testo; può assumere i seguenti valori: J_NOJUST = Nessuna giustificazione J_LEFT = Giustificazione a sinistra J_CENTER = Al centro J_RIGHT = Giustificazione a destra J_JUST = Giustificazione a pacchetto
JustificationLength *as Integer*	Lunghezza nella quale effettuare la giustificazione (dovrebbe essere sempre maggiore o uguale alla lunghezza del testo da giustificare)
Pattern *as Integer*	Codice ASCII del carattere utilizzato per la giustificazione

Se *X* = 0, la funzione entra in modo emulazione, nel quale nulla viene stampato a video ma sono calcolate e restituite solo le righe necessarie alla stampa. Vedi anche HPrint.

ScreenClear (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer,*

ByVal Bottom as Integer)

void ScreenClear(short Left, short Top, short Right, short Bottom);

Cancella l'area del video compresa tra (*Left, Top*) e (*Right, Bottom*). I colori utilizzati corrispondono ai valori delle proprietà ForegroundColor e BackgroundColor, il carattere di cancellazione corrisponde alla proprietà Pattern.

ScreenRestore (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer,*
ByVal Bottom as Integer, ByVal Screen as String)

void ScreenRestore(short Left, short Top, short Right, short Bottom, LPCTSTR ScreenBuffer);

Ripristina il blocco video *Screen* (ottenuto da una precedente chiamata a ScreenSave) alle coordinate specificate. Le dimensioni dell'area di destinazione devono corrispondere esattamente, per larghezza e lunghezza, a quelle dell'area salvata con ScreenSave; le coordinate, invece, possono anche essere diverse.

ScreenSave (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer,*
ByVal Bottom as Integer) *as String*

CString ScreenSave(short Left, short Top, short Right, short Bottom);

Restituisce una stringa che rappresenta il blocco video (testo ed attributi) dell'area compresa tra (*Left, Top*) e (*Right, Bottom*); quest'area può successivamente essere ripristinata attraverso ScreenRestore.

ScrollHorizontally (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer,*
ByVal Bottom as Integer, ByVal Columns as Integer)

void ScrollHorizontally(short Left, short Top, short Right, short Bottom, short Columns);

Scrolla orizzontalmente l'area indicata, del numero di colonne pari a *Columns*: se quest'ultimo è positivo, lo scorrimento avviene verso sinistra; se è negativo, lo scorrimento è verso destra.

ScrollVertically (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer,*
ByVal Bottom as Integer, ByVal Rows as Integer)

void ScrollVertically(short Left, short Top, short Right, short Bottom, short Rows);

Scrolla verticalmente l'area indicata, del numero di righe pari a *Rows*: se quest'ultimo è positivo, lo scorrimento avviene verso l'alto; se è negativo, lo scorrimento è verso il basso.

SettingsRestore (*ByVal Settings as String*)

void SettingsRestore(LPCTSTR SavedSettings);

Ripristina i valori di tutte le proprietà del controllo xConsole®; *Settings* deve essere stata precedentemente ottenuta mediante l'invocazione del metodo SettingsSave.

SettingsSave () *as String*

CString SettingsSave();

Restituisce una stringa che contiene tutti i valori delle proprietà del controllo xConsole®; in questo modo, è possibile effettuare qualsiasi variazione richiesta (inclusa l'invocazione ricorsiva di metodi), purché si abbia l'accortezza di ripristinare poi i valori originali delle proprietà attraverso SettingsRestore.

Shadow (*ByVal Left as Integer, ByVal Top as Integer, ByVal Right as Integer, ByVal Bottom as Integer*)

void Shadow(short Left, short Top, short Right, short Bottom);

Disegna un'ombra per l'area compresa tra (*Left*, *Top*) e (*Right*, *Bottom*); il tipo dell'ombra disegnata dipende dal valore della proprietà ShadowMode, che può essere:

NO_SHADOW = nessuna ombra
SHADOW_LEFT = ombra a sinistra
SHADOW_RIGHT = ombra a destra

ShutdownConsole ()

void ShutDownConsole();

Chiude gli handle e libera la console allocata da InitConsole; dovrebbe essere l'ultimo metodo invocato prima della terminazione del programma. Di solito, non è necessario invocarlo: la fine del programma comporta la chiusura automatica di tutti gli handle associati al processo, fra cui (appunto) la console.

TextBox (*ByVal Tag as Integer*) *as Boolean*

BOOL TextBox(long Tag);

TextBoxXY (*ByVal X as Integer, ByVal Y as Integer, ByVal Tag as Integer*) *as Boolean*

BOOL TextBoxXY(short X, short Y, long Tag);

Permette di modificare un buffer rettangolare di testo; la prima forma utilizza le coordinate correnti, la seconda quelle specificate. Ritorna True se l'utente ha confermato l'editing (Enter oltre l'ultima riga oppure Ctrl+Enter), False se è uscito con Esc. Il comportamento del metodo è influenzato dalle seguenti proprietà:

TextBoxDefault *as String*	Valore iniziale del buffer (al ritorno contiene quanto editato); questa proprietà è aggiornata anche se l'utente preme Esc
TextBoxColumns *as Integer*	Numero di colonne della finestra di editing
TextBoxRows *as Integer*	Numero di righe della finestra di editing
TextBoxStartPosition *as Integer*	Posizione del cursore nel buffer rettangolare; questa proprietà è aggiornata al termine, anche se l'utente preme Esc
InsertMode *as Boolean*	Stato di inserimento: se True, ogni carattere digitato viene inserito, spostando in avanti gli eventuali caratteri seguenti; se False, il carattere sovrascrive quanto esistente; questa proprietà è aggiornata al termine, anche se l'utente preme Esc
TextBoxBackColor *as Integer* TextBoxForeColor *as Integer*	Colori di sfondo e primo piano utilizzati per la finestra di editing
SilentMode *as Boolean*	Se True, nessuna segnalazione acustica sarà emessa in caso di errore (sarà comunque invocato l'evento SoundRequest)

Il parametro *Tag* determina la reazione del metodo all'introduzione di dati da parte dell'operatore; se è zero, il comportamento di default è il seguente:

Ctrl+Enter, Enter sull'ultima riga	conferma editing

Esc	annulla editing
Tasti cursore	navigano l'area
Home, End	inizio/fine riga
Ctrl+Home, Ctrl+End	inizio/fine buffer
Ctrl+Sinistra, Ctrl+Destra	parola precedente/successiva
Delete	cancella il carattere attuale e sposta all'indietro il testo
Backspace	cancella il carattere precedente e sposta indietro il testo
Enter	inserisce spazi fino alla fine della riga (porta a capo ciò che segue) se l'inserimento è attivo; introduce una fine paragrafo al termine della riga corrente (se possibile) altrimenti
Ctrl+Y	cancella la riga attuale
Ins	grazie alla procedura INSTOGGLE, cambia la forma del cursore e lo stato di inserimento
Ctrl+B	mostra/nasconde i contrassegni di paragrafo
Ctrl+W	porta a capo della prossima riga la parola a sinistra (wrap)
Ctrl+E	porta il cursore alla fine della parola editata
Ctrl+N	azzera il buffer e riporta il cursore all'inizio
Ctrl+S	salva il buffer come checkpoint
Ctrl+L	ripristina all'ultimo checkpoint

I contrassegni di paragrafo sono gestiti grazie all'introduzione del simbolo CHR(255) nel testo, che risulta comunque invisibile in modalità console. Dopo l'esecuzione del metodo, ricordarsi di sostituire opportunamente questo carattere con spazio, in quanto in modalità grafica questo carattere è invece visibile.

Se *Tag* è diverso da zero, ad ogni tasto premuto viene generato il seguente evento:

TextBoxKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)

dove:

KeyAscii *as Integer*	contiene il tasto introdotto dall'utente; può essere aggiornato per simulare la pressione di un diverso tasto e impostato a 0 per scartarlo
Action *as String*	determina l'azione richiesta in risposta dal metodo; può essere aggiornato con uno dei valori indicati nella tabella più sotto
Tag *as Integer*	numero identificativo della TextBox

Il valore di *Tag* permette di conoscere qual'è la TextBox attiva, potendo in questo modo adottare comportamenti diversi a seconda delle necessità.

I valori possibili per *Action* sono i seguenti:

TEXTBOX_ACCEPT	Accetta il carattere ed inseriscilo nella stringa
TEXTBOX_UPDATE	Buffer aggiornato, riprendi editing
TEXTBOX_UPDATEANDCONFIRM	Buffer aggiornato, accetta
TEXTBOX_ABORT	Annulla l'input
TEXTBOX_CONFIRM	Conferma l'input
TEXTBOX_DISCARD	Scarta il carattere

TEXTBOX_LEFT	Sposta il cursore a sinistra
TEXTBOX_RIGHT	Sposta il cursore a destra
TEXTBOX_HOME	Sposta il cursore all'inizio
TEXTBOX_END	Sposta il cursore alla fine
TEXTBOX_DELLINE	Cancella la riga attuale
TEXTBOX_BOL	Inizio riga
TEXTBOX_EOL	Fine riga
TEXTBOX_PWORD	Parola precedente
TEXTBOX_NWORD	Parola successiva
TEXTBOX_PARSIGN	Mostra/nasconde i contrassegni di paragrafo
TEXTBOX_WRAP	Porta a capo della prossima riga la parola a sinistra (wrap)
TEXTBOX_EOW	Porta il cursore alla fine della parola editata
TEXTBOX_CLEAR	Azzera il buffer e porta il cursore all'inizio della stringa
TEXTBOX_CHKRESTORE	Ripristina all'ultimo checkpoint
TEXTBOX_CHECKPOINT	Checkpoint

ThumbElevator (*ByVal Current as Long, ByVal Total as Long, ByVal Column as Integer, ByVal FirstRow as Integer, ByVal LastRow as Integer, ByVal ForeColor as Integer, ByVal BackColor as Integer, ByRef LastPosition as Integer*)

void ThumbElevator(long Current, long Total, short Column, short FirstRow, short LastRow, short ForeColor, short BackColor, short* LastPosition);

Disegna un indicatore di posizione a colonna da (*Column*, *FirstRow*) a (*Column*, *LastRow*), con il colore *ForeColor* / *BackColor*. *Current* è il valore corrente (inizia da 0), *Total* è il valore massimo; *LastPosition* mantiene la posizione del cursore e viene aggiornato al ritorno; se –1, forza il ritracciamento dell'intero indicatore (si consiglia di utilizzarlo come valore iniziale).

Tone (*ByVal Frequency as Long, ByVal Duration as Long*)

void Tone(long Frequency, long Duration);

Emette un segnale acustico della frequenza e durata richiesti; *Frequency* deve essere espresso in Hertz, *Duration* in millisecondi.
Se la proprietà SilentMode è True, non viene prodotto alcun suono, ma viene generato l'evento:

SoundRequest (*ByVal Frequency as Long, ByVal Duration as Long*)

In questo modo, è possibile dare una segnalazione alternativa, oppure utilizzare sequenze di escape particolari per far produrre un suono al terminale remoto.

Proprietà di xConsole®

Di seguito sono descritte tutte le proprietà del controllo xConsole®, incluse le loro interazioni (sottolineate attraverso un prefisso comune).

I metodi sono indicati in **BLU**, le proprietà in **ROSSO**.

Le costanti sono indicate sempre attraverso identificatori mnemonici; nel modulo XCONSOLE.BAS e nel file header XCONSOLE.H compaiono i corrispondenti valori effettivi.

La sintassi indicata segue quella di Visual Basic; per Visual C++ valgono le seguenti conversioni di tipo:

Tipo Visual Basic	Tipo Visual C++
Boolean	BOOL
Integer	short o short * (se passato per riferimento)
Long	long o long * (se passato per riferimento)
String	LPCTSTR (parametro nei metodi) CString (valore di proprietà) BSTR (valore ritornato da un metodo)

Tutte le proprietà del controllo sono assegnate in Visual C++ invocando funzioni il cui nome corrisponde a quello della proprietà con prefisso “Set”; tali funzioni hanno come unico parametro il valore da assegnare alla proprietà; ad esempio MenuOptions si imposta con:

```
SetMenuOptions(opzioni);
```

Analogamente, per prelevare il valore di una proprietà si utilizza una pseudo-funzione avente come prefisso “Get”:

```
opzioni = GetMenuOptions();
```

Tutti i metodi aventi la variante con suffisso “XY” sono citati solo nella loro variante semplice.
Tutte le proprietà sono di lettura/scrittura.

Elenco alfabetico delle proprietà (si rimanda ai metodi per informazioni più dettagliate):

AlertBackColor *as Integer*	Colore di sfondo per Alert
AlertButtonBackColor *as Integer*	Colore di sfondo dei pulsanti per Alert
AlertButtonForeColor *as Integer*	Colore di primo piano dei pulsanti per Alert
AlertButtons *as String*	Testo dei pulsanti per Alert
AlertCurrentButton *as Integer*	Indice del pulsante corrente per Alert
AlertForeColor *as Integer*	Colore di primo piano per Alert
AlertFrameForeColor *as Integer*	Colore di primo piano della cornice per Alert
AlertText *as String*	Testo da stampare nell’Alert
BackgroundColor *as Integer*	Colore di sfondo per i metodi di stampa diretta (SPrint, HPrint)
ConsoleTitle *as String*	Titolo della finestra (quando il programma non viene eseguito a tutto schermo)

CursorType *as Integer*	Tipo cursore; può valere: CUR_OFF = cursore nascosto CUR_BIG = cursore a blocco CUR_SMALL = cursore a trattino La variazione della proprietà comporta l'immediata modifica del cursore.
DateType *as Integer*	Formato della data, per la validazione automatica nei metodi InputString; può valere: DATE_US = mese/giorno/anno DATE_EUROPE = giorno/mese/anno DATE_JAPAN = anno/mese/giorno
Epoch *as Integer*	Valore dell'epoca, per la validazione automatica nei metodi InputString. L'epoca determina come devono essere interpretati gli anni nelle date contratte, cioé quelle date in cui per l'anno sono utilizzate solo due cifre; in questo caso, se decine ed unità sono inferiori alle decine ed unità dell'anno di riferimento in Epoch, viene assunto come riferimento il secolo successivo, altrimenti il secolo attuale.
ForegroundColor *as Integer*	Colore di primo piano per i metodi di stampa diretta (SPrint, HPrint)
Frame *as Boolean*	Indica se una cornice debba essere aggiunta ai box in tutti i metodi che la prevedono (es. Alert, List, OSD, ecc.)
Frame3D *as Boolean*	Indica se la cornice debba avere un aspetto a rilievo (due lati adiacenti hanno una colorazione più scura rispetto agli altri due lati)
FrameBackColor *as Integer*	Colore di sfondo per Box
FrameChars *as String(8)*	Stringa di 8 bytes che rappresentano in senso orario i caratteri da utilizzare per disegnare la cornice, partendo dall'angolo superiore sinistro ed andando in senso orario. Il default, dopo InitConsole, permette di tracciare cornici a tratto singolo (FRAME_SINGLE) sfruttando i caratteri semigrafici. Le costanti, FRAME_DOUBLE, FRAME_SNGDOU, FRAME_DOUSNG e FRAME_DOTS possono essere utilizzate in alternativa.
FrameForeColor *as Integer*	Colore di primo piano per Box
FullScreen *as Boolean*	Indica se l'applicazione è in esecuzione in una finestra (False) oppure a video intero (True)
InputBackColor *as Integer*	Colore di sfondo per InputString
InputCodePage *as Long*	CodePage utilizzata per l'input
InputDefault *as String*	Valore della stringa editata in InputString
InputForeColor *as Integer*	Colore di primo piano per InputString
InputMaxLength *as Integer*	Massima lunghezza della stringa per InputString
InputPicture *as String*	Formato di validazione per InputString
InputStartPos *as Integer*	Posizione iniziale del cursore per InputString
InputWindowLength *as Integer*	Lunghezza della finestra di editing per InputString

InputWindowOffset *as Integer*	Spostamento all'interno della finestra di editing per InputString
InsertMode *as Boolean*	Indica la modalità inserimento/sovrascrittura, usata da InputString e TextBox
Justification *as Integer*	Giustificazione del testo per i metodi di stampa diretta (SPrint, HPrint); può assumere i seguenti valori: J_NOJUST = Nessuna giustificazione J_LEFT = Giustificazione a sinistra J_CENTER = Al centro J_RIGHT = Giustificazione a destra J_JUST = Giustificazione a pacchetto
JustificationLength *as Integer*	Lunghezza di giustificazione per i metodi di stampa diretta (SPrint, HPrint)
KeyLast *as Long*	Codice dell'ultimo tasto letto con KeyInput; proprietà di lettura/scrittura
LineCharsHV *as String(2)*	Stringa di due caratteri, utilizzata dal metodo LineFromTo; il carattere 1 viene utilizzato per disegnare le linee orizzontali, il carattere 2 le linee verticali
ListColumns *as Integer*	Numero di colonne occupate dalla lista di opzioni per il metodo List
ListCurrentLine *as Integer*	Linea corrente per il metodo List
ListCurrentOption *as Integer*	Opzione corrente per il metodo List
ListFrameForeColor *as Integer*	Colore di primo piano della cornice per il metodo List
ListMap *as String*	Mappa di selezione delle opzioni per il metodo List
ListMultiSelect *as Boolean*	Indica se il metodo List debba ammettere la selezione multipla
ListOptions *as String*	Elenco delle opzioni per il metodo List
ListRows *as Integer*	Numero di righe occupate dalla lista di opzioni per il metodo List
ListSelection *as Integer*	Carattere di spunta opzione per il metodo List
ListTitle *as String*	Titolo della lista opzioni per il metodo List
ListWindowOffset *as Integer*	Offset di stampa delle opzioni (numero di caratteri da saltare all'inizio di ciascuna opzione) per il metodo List
MaxCol *as Integer*	Numero di colonne dello schermo
MaxRow *as Integer*	Numero di righe dello schermo
MenuCurrentOption *as Integer*	Indice della voce di menu corrente per Menu
MenuFrameForeColor *as Integer*	Colore di primo piano della cornice per Menu
MenuOptions *as String*	Lista delle opzioni (con rispettiva descrizione) per Menu
MenuUnselectable *as String*	Mappa delle opzioni non selezionabili per Menu
OffsetX *as Integer*	Spostamento orizzontale predefinito
OffsetY *as Integer*	Spostamento verticale predefinito
OutputCodePage *as Long*	CodePage utilizzata per la scrittura a video (normalmente significativa solo quando la proprietà FullScreen è True)

Pattern *as Integer*	Carattere utilizzato per tutte le operazioni di riempimento/cancellazione
ScoreboardBackColor *as Integer*	Colore di sfondo per le descrizioni (metodo Menu)
ScoreboardForeColor *as Integer*	Colore di primo piano per le descrizioni (metodo Menu)
ScoreboardJustification *as Integer*	Tipo giustificazione per le descrizioni (metodo Menu)
ScoreboardLength *as Integer*	Lunghezza di giustificazione per le descrizioni (metodo Menu)
ScoreboardStatus *as Boolean*	Indica lo stato di abilitazione per la stampa delle descrizioni (metodo Menu)
ScoreboardX *as Integer*	Ascissa per la stampa delle descrizioni (metodo Menu)
ScoreboardY *as Integer*	Ordinata per la stampa delle descrizioni (metodo Menu)
SelectedBackColor *as Integer*	Colore di sfondo per le opzioni selezionate (metodi List, Menu)
SelectedForeColor *as Integer*	Colore di primo piano per le opzioni selezionate (metodi List, Menu)
ShadowMode *as Integer*	Tipo di ombra, per dare evidenza alla cornice; i possibili valori sono: NO_SHADOW = nessuna ombra SHADOW_LEFT = ombra a sinistra SHADOW_RIGHT = ombra a destra L'ombra viene aggiunta automaticamente quando viene disegnato un Box (implicitamente o esplicitamente)
SilentMode *as Boolean*	Indica se il metodo Tone debba far emettere un suono oppure generare l'evento SoundRequest
TextBoxBackColor *as Integer*	Colore di sfondo per il metodo TextBox
TextBoxColumns *as Integer*	Numero di colonne per il metodo TextBox
TextBoxDefault *as String*	Valore del buffer di editing per TextBox
TextBoxForeColor *as Integer*	Colore di primo piano per il metodo TextBox
TextBoxRows *as Integer*	Numero di righe per il metodo TextBox
TextBoxStartPosition *as Integer*	Posizione iniziale nel buffer di editing per TextBox
ThumbElevatorChars *as String(4)*	Stringa contenente i caratteri da utilizzare per disegnare l'indicatore a colonna: 1 – terminatore alto 2 – terminatore basso 3 – sfondo 4 – indicatore
TitleBackColor *as Integer*	Colore di sfondo per il titolo della lista opzioni (metodo List)
TitleForeColor *as Integer*	Colore di primo piano per il titolo della lista opzioni (metodo List)
UnselectableBackColor *as Integer*	Colore di sfondo per le voci non selezionabili del metodo Menu
UnselectableForeColor *as Integer*	Colore di primo piano per le voci non

	selezionabili del metodo Menu
UnselectedBackColor *as Integer*	Colore di sfondo per le opzioni non selezionate (metodi List, Menu)
UnselectedForeColor *as Integer*	Colore di primo piano per le opzioni non selezionate (metodi List, Menu)
X *as Integer*	Coordinata X del cursore; proprietà di lettura/scrittura (l'aggiornamento comporta l'immediato spostamento del cursore)
Y *as Integer*	Coordinata Y del cursore; proprietà di lettura/scrittura (l'aggiornamento comporta l'immediato spostamento del cursore)

Eventi di xConsole®

Di seguito sono descritti tutti gli eventi del controllo xConsole®.
I metodi sono indicati in **BLU**, le proprietà in **ROSSO**, gli eventi in **VERDE**.
La sintassi indicata segue quella di Visual Basic; per Visual C++ valgono le seguenti conversioni di tipo:

Tipo Visual Basic	Tipo Visual C++
Boolean	BOOL
Integer	short o short * (se passato per riferimento)
Long	long o long * (se passato per riferimento)
String	LPCTSTR (parametro nei metodi) CString (valore di proprietà) BSTR (valore ritornato da un metodo)

In Visual C++ è anche necessario introdurre un gestore di eventi, come specificato nel capitolo che spiega l'utilizzo del controllo in questo linguaggio.

Eventi legati alla pressione di un tasto

Ogni tasto acquisito genera il seguente evento:

KeyPress(*ByRef KeyAscii as Integer*)

KeyAscii contiene il tasto introdotto dall'utente; può essere aggiornato per simulare la pressione di un diverso tasto o impostato a 0 per scartarlo.

Gli eventi seguenti hanno tutti la stessa struttura; sono invocati dai rispettivi metodi quando *Tag* è diverso da zero:

AlertKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)
InputStringKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)
ListKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)
MenuKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)
TextBoxKeyPress(*ByRef KeyAscii as Integer, ByRef Action as Integer, ByVal Tag as Long*)

dove:

KeyAscii *as Integer*	Contiene il tasto introdotto dall'utente; può essere aggiornato per simulare la pressione di un diverso tasto e impostato a 0 per scartarlo
Action *as String*	Determina l'azione richiesta in risposta dal metodo; può essere aggiornato con uno dei valori indicati nella tabella più sotto
Tag *as Integer*	Numero identificativo del metodo che ha generato l'evento

Altri eventi

SoundRequest (*ByVal Frequency as Long, ByVal Duration as Long*)

Questo evento è generato dal metodo Tone quando la proprietà SilentMode è True; permette di sostituire la routine di generazione del suono di Tone con un altro sistema di segnalazione a scelta dell'utente.

Espressioni regolari

xConsole® supporta nei metodi InputString due tipi di espressioni regolari: estese e semplici.

Espressioni regolari estese

Le espressioni regolari estese sono più potenti ma anche più complesse da utilizzare; esse consistono in una stringa di caratteri in cui alcuni sono interpretati letteralmente, mentre altri sono caratteri di controllo con significato speciale.

Ecco una breve spiegazione sul loro uso:

a) una '\' seguita da un carattere singolo implica la corrispondenza con quel carattere;
b) il carattere '^' corrisponde all'inizio della stringa; il carattere '$' corrisponde alla fine della stringa;
c) un '.' corrisponde a qualsiasi carattere;
d) un carattere singolo senza significati speciali implica la corrispondenza con il carattere medesimo;
e) una stringa racchiusa in parentesi [quadre] implica la corrispondenza con uno qualsiasi dei caratteri nella stringa. Intervalli di caratteri ASCII possono essere abbreviati come 'a-z0-9'. Una parentesi ']' isolata può occorrere solo come primo carattere della stringa con l'espressione regolare. Un '-' letterale deve essere posizionato dove non può essere scambiato per un indicatore di intervallo. Se il primo carattere è '^' allora ogni carattere non corrispondente all'espressione sarà accettato;
f) un'espressione regolare seguita da '*' corrisponde ad una sequenza di 0 o più corrispondenze con l'espressione regolare;
g) un'espressione regolare seguita da '+' corrisponde ad una sequenza di 1 o più corrispondenze con l'espressione regolare;
h) un'espressione regolare seguita da '?' corrisponde ad una sequenza di 0 o 1 corrispondenze con l'espressione regolare;
i) due espressioni regolari adiacenti (concatenate) implicano la corrispondenza con la prima seguita dalla corrispondenza con la seconda;
j) due espressioni regolari separate da '|' implicano la corrispondenza con la prima o la seconda;
k) un'espressione regolare tra (parentesi) implica la corrispondenza con il suo contenuto.

L'ordine di precedenza degli operatori allo stesso livello di parentesi è (da massima precedenza a minima precedenza):

[] *+? concatenazione |

Alcuni esempi di espressioni regolari estese:

"^a"	Accetta una stringa che inizia per 'a'
"^mele"	Accetta una stinga che inizia per 'mele'
"a$"	Accetta una stringa che finisce per 'a'
"arance$"	Accetta una stringa che finisce per 'arance'
"m..a"	Accetta una stringa che contiene una parola di 4 lettere che inizia per 'm' e finisce per 'a' (es. 'mela', 'meta' ma non 'manna')
"[ab]"	Accetta una stringa che contiene 'a' o 'b'
"[^ab]"	Accetta una stringa che contiene un carattere diverso da 'a' e 'b'
"^[0-3][0-9]/[0-1][0-	Accetta una data (tipo "30/12/97")

9]/[0-9][0-9]$"	
"c(a|o)sa"	Accetta una stringa che contiene la parola 'casa' o 'cosa' (non 'ca' o 'osa')
"principii?"	Accetta una stringa contenente 'principi' o 'principii'
"^[0-9]*$"	Accetta una stringa vuota oppure un numero con le sole cifre '0'-'9'
"^[a-zA-Z]+[a-zA-Z0-9]*$"	Accetta un nome di identificatore (inizia per una lettera, può contenere solo lettere o numeri, è lungo almeno un carattere)
"^(salve)|(arrivederci)$"	Accetta le due stringhe 'salve' e 'arrivederci'

Espressioni regolari semplici

Le espressioni regolari semplici sono più facili da utilizzare rispetto a quelle estese; a differenza di queste ultime, presentano solo due caratteri speciali:

1. '*' sostituisce zero, uno o più caratteri qualsiasi
2. '?' sostituisce un carattere qualsiasi

Alcuni esempi di espressioni regolari semplici:

"*c?sa*"	accetta una stringa che contiene la lettera 'c' e la stringa 'sa' separate da un unico carattere (ad. esempio 'casa', 'cosa')
"c*"	accetta una stringa che inizia per 'c'
"*a"	accetta una stringa che finisce per 'a'
"???"	accetta una stringa di tre caratteri
"*uno*due*tre*"	accetta una stringa che presenta le tre stringhe 'uno', 'due', 'tre' in quest'ordine

Licenza d’uso

Leggere attentamente prima di installare il software.

Installando il software accluso confermate di aver letto, compreso e accettato i termini e le condizioni di questa Licenza d’uso del software. Se non accettate i termini di questa Licenza, ritornate i dischi, eventuali contenitori e documentazione al rivenditore per ottenere un completo rimborso della cifra pagata per l’acquisto.

LICENZA SOFTWARE

Questo è un Contratto legale stipulato tra voi (un individuo o un’entità) e la Simone Zanella Productions (SZP) per stabilire i termini e le condizioni d’uso del Software allegato. Aggiornamenti del Software saranno anch’essi soggetti ai termini ed alle condizioni di questo Contratto. Questo Contratto è in vigore finché terminato dalla distruzione del Software e di tutta la documentazione fornita con il pacchetto, assieme a tutte le copie, tangibili o intangibili. In questo Contratto, il termine “uso” significa il caricamento del Software nella RAM, come anche l’installazione su hard disk o altra periferica di memorizzazione.
Il Software è di proprietà della Simone Zanella Productions ed è protetto dalle leggi italiane sul diritto d’autore e dai trattati internazionali sul copyright. Dovete considerare il Software come qualsiasi altro materiale protetto dal diritto d’autore. Il prezzo d’acquisto del Software vi concede una licenza non-esclusiva per l’uso di una copia del Software su un singolo computer. SZP mantiene titolo e proprietà sul Software.
Potete fare una copia del Software solo a scopo di archivio. Non potete usare i dischi o il file licenza su un computer diverso da quello di installazione, né permettere a terze parti di utilizzare il Software attraverso time-sharing, rete o qualsiasi altra forma di partecipazione multi-utente.
Non potete noleggiare, vendere, prestare, sub-licenziare o condividere in qualsiasi modo il Software con una terza parte, e nemmeno trasferire questa Licenza senza un permesso scritto rilasciato da SZP. Non potete decompilare, disassemblare, modificare oppure applicare le tecniche di *reverse-engineering* al Software. Potete altresì distribuire applicativi compilati, realizzati utilizzando il software, purché con essi non venga mai distribuito il file licenza.
Se non rispettate uno qualsiasi dei termini e delle condizioni di questo Contratto, la Licenza sarà immediatamente terminata e dovrete ritornare immediatamente alla SZP il Software, i dischetti e la documentazione di questo pacchetto, assieme a tutte le copie di back-up. Quanto previsto da questo Contratto protegge i diritti proprietari di SZP anche dopo la terminazione.

RESPONSABILITÀ LIMITATA

Il Software e la documentazione sono venduti **COME SONO**. Vi assumete la responsabilità della scelta di questo Software per raggiungere i risultati desiderati e per l’installazione, l’uso ed i risultati ottenuti dal Software. SZP non fornisce alcuna affermazione o garanzia riguardo al Software e alla documentazione, incluse, ma non solo, le garanzie implicite di commerciabilità o adattabilità ad un uso specifico. SZP non sarà responsabile per errori od omissioni contenute nel Software o nei manuali, per qualsiasi interruzione del servizio, perdita di affari o profitti e/o per i danni incidentali o conseguenti in connessione con la fornitura o il funzionamento di questi materiali.

GARANZIA LIMITATA

Per un periodo di ventiquattro (24) mesi dalla data di acquisto, la SZP garantisce l’acquirente originale che il disco sul quale il Software è registrato è libero da difetti nei materiali o nelle lavorazioni quando soggetto ad uso e servizio normali. Se durante questi ventiquattro (24) mesi dovesse evidenziarsi un difetto, il disco sarà rimpiazzato gratuitamente dopo la restituzione ad SZP. Se un difetto dovesse evidenziarsi scaduto il periodo di garanzia, la SZP si riserva il diritto di far

pagare un costo per l’intervento. La SZP si riserva il diritto di rifiutare richieste di sostituzione reiterate. Questa Garanzia Limitata vi fornisce dei diritti specifici e potete anche avere altri diritti che variano da Paese a Paese. Alcuni Paesi non permettono la limitazione o l’esclusione di garanzie implicite o di danni conseguenti, per cui le limitazioni o esclusioni indicate potrebbero non riferirsi a voi.

Questo testo è il completo ed esclusivo Contratto tra voi e la SZP, il quale sostituisce qualsiasi proposta o patto precedente, orale o scritto, e ogni altra comunicazione tra noi riguardo la materia oggetto di questo Contratto. Questo Contratto sarà interpretato e governato dalle leggi dello Stato Italiano ed ogni contestazione sarà discussa al foro di Venezia. Se alcuni dei punti di questo contratto risultano inapplicabili, essi non inficieranno la validità dell’insieme di questo Contratto, che resterà valido ed applicabile in base ai suoi termini.

INDICE ANALITICO