La classe TPrinter
con uno sguardo al nuovo ObjectWindows

In questo e nel prossimo numero vedremo l'implementazione della classe TPrinter, *mediante la quale sarà possibile condurre operazioni di stampa in modo testo sotto DOS prescindendo dalle peculiarità della stampante fisicamente disponibile. Vedremo anche come e dove modificare il codice per estendere le funzionalità dell'impianto proposto, tenendo anche conto dei contributi di alcuni abbonati a MC-link*

di Sergio Polini

Il nostro cammino era iniziato da una discussione dell'uso di una stampante sotto Windows, allo scopo di trarne ispirazione per poter effettuare anche sotto DOS operazioni di stampa in modo indipendente dalle peculiarità di ogni singola stampante.

Nell'ambito di quella discussione, avevamo avuto modo di apprezzare che, nonostante l'indubbio beneficio dell'indipendenza dall'hardware, l'uso di una stampante sotto Windows non è poi così semplice; avevo in animo, in effetti, di far seguire alla serie di articoli dedicata alle stampe sotto DOS un'analoga serie dedicata ad unit che agevolassero l'uso di una stampante sotto Windows.

La Borland mi ha anticipato: il Borland Pascal 7.0, infatti, include una nuova versione della gerarchia di classi ObjectWindows in cui compaiono anche le classi *TPrinter, TPrintOut* e *TWindowPrintout*.

Grazie ad esse, per stampare sono sufficienti tre operazioni: la costruzione di un'istanza di *TPrinter*, normalmente nell'ambito del constructor della finestra principale dell'applicazione; la creazione di un'istanza di *TPrintOut* o di una classe da questa derivata, come appunto *TWindowPrintout*, che può essere usata per stampare il contenuto di una finestra; infine si stampa, operazione che, nei casi ordinari, può ridursi ad una semplice chiamata del metodo *TPrinter.Print* con un parametro che sia un puntatore all'istanza di *TWindowPrintout*. La classe *TPrinter* ha anche un metodo *Setup*, grazie al quale si ottiene l'apertura della dialog box per la scelta della stampante da utilizzare e per la modifica della sua impostazione. La figura 1 permette di apprezzare quanto sia semplice l'uso delle nuove classi.

Lavorando con Turbo Vision, invece, si deve ancora provvedere a tutti i dettagli. Può quindi valere la pena di proseguire nell'esame della unit TVPRINT.

Inizializzazione della stampante e inizio della stampa

La classe *TPrinter* di TVPRINT dispone sia di un constructor e di un destructor, sia dei metodi *Start, Reset* e *Finish*; i primi due (figura 2) presiedono alla costruzione e successiva distruzione di un oggetto che sia istanza della classe, gli altri tre (figura 3) gestiscono l'inizio e la fine di una fase di stampa.

Si sarebbe forse potuto prevedere l'uso di un constructor per l'inizializzazione di ogni singola fase di stampa, ma, così facendo, si sarebbe dovuto anche creare un'istanza di *TPrinter* per operazioni come l'impostazione della stampante o la scelta del tipo di carattere da utilizzare (se operata prima di dare avvio alla stampa).

Il constructor, quindi, si limita a verificare la presenza di una stampante di default, mediante ricerca di una variabile TVPRN nell'environment. Se la ricerca ha esito positivo, si legge la definizione della stampante mediante la funzione *ReadPrinterInfo*; in caso contrario, si assegna una stringa nulla al campo *Name* della variabile d'istanza PD.

In ogni caso, si assegnano valori di default ad altre variabili d'istanza, quali *Font* (zero, per il primo tipo di carattere supportato), *TabWidth* (0.5, per un intervallo di tabulazione di mezzo pollice), *LineSpacing* (1/6, per un'interlinea di 6 righe per pollice), e *OkToPrint* (FALSE; la variabile potrà assumere il valore TRUE solo dopo l'esito positivo delle azioni di cui si incarica il metodo Start e varrà nuovamente FALSE in caso di errori di I/O).

Ricordo che *LineSpacing*, pur essendo una variabile, mantiene sempre lo stesso valore. Potrebbe tuttavia diventare una vera e propria variabile se in PRSETUP si prevedesse l'acquisizione di una sequenza di controllo per l'impostazione dell'interlinea.

Il campo *PD.Name* può venire utilizzato da un'applicazione per verificare l'avvenuta corretta creazione di un'istanza di *TPrinter* corrispondente ad una stampante di default; quando abbiamo esaminato il programma DEMO nel numero di novembre, abbiamo visto una funzione *PrinterSelected* che provvedeva ad una tale verifica. In quella occasione, abbiamo anche chiarito che la funzione non è un metodo di *TPrinter* in quanto è opportuno che proponga all'utente una determinata azione dipendente dalla struttura dell'applicazione, quale la scelta di un'opzione «Imposta stampante» in un menu «File».

Il metodo *Start* provvede in primo luogo all'apertura del dispositivo di output, utilizzando, nel caso di uscita su file, la classe *TPrintFileDialog* che abbiamo visto la volta scorsa. Se non si incontrano ostacoli, dopo aver assegnato il valore TRUE alla variabile *OkToPrint*, si inizializzano altre variabili e si prepara la visualizzazione della dialog box che consentirà di interrompere una stampa in corso premendo Esc.

La pressione del tasto Esc durante una stampa provoca l'assegnazione di TRUE alla variabile *UserAbort*, che viene pertanto inizializzata con FALSE. Altre variabili, come *BFace* e *ULine*, vengono inizializzate mediante i metodi *BoldFace* e *Underline*, chiamati subito dopo il metodo *Reset* e intesi come completamento di questo. Con ciò si ottiene che la sequenza di controllo specificata per il reset della stampante non deve contenere anche i comandi per la disattivazione di grassetto e sottolineato, ma può essere limitata ad altre azioni (ad esempio, impostare un orientamento orizzontale o verticale del foglio di carta). Segue il metodo *SetFont*, mediante il quale si assegna alla variabile *CharWidth* un valore corrispondente al tipo di carattere selezionato; questo sarà o il primo, a seguito della inizializzazione della variabile *Font* a zero operata dal constructor, oppure un altro

eventualmente scelto mediante il metodo *SelectFont*.

Anche qui ci imbattiamo in una possibile estensione delle potenzialità della unit TVPRINT. *CharWidth* infatti, come già accennato nel numero di dicembre, potrebbe essere una funzione che, passato come parametro un carattere, rendesse la sua ampiezza. Ciò consentirebbe di usare tipi di carattere proporzionali, ma richiederebbe una modifica della unit PRSETUP atta a consentire, con una apposita dialog box, l'acquisizione dell'ampiezza di ogni singolo carattere stampabile per tutti i font proporzionali.

L'esecuzione dei metodi chiamati da *Start* comporta l'invio di dati al dispositivo di output e, quindi, potrebbe fallire, con conseguente assegnazione di FALSE alla variabile *OkToPrint*. Se questa vale ancora TRUE, si procede alla inizializzazione delle variabili *MaxX, MaxY* (le dimensioni dell'area di stampa), *PosX* e *PosY* (la posizione della testina di stampa; tutte le misure si intendono in pollici) e, quindi, alla preparazione della dialog box per l'interruzione delle stampe. Si tratta di una dialog box non modale (aperta con *Insert* invece che con *ExecView*) in quanto, in caso contrario, la sua visualizzazione comporterebbe l'arresto della stampa fino alla sua chiusura; per evitare comunque che l'utente azioni altri comandi mentre la dialog box è visualizzata, i comandi che risultano abilitati vengono salvati nella variabile *SaveCommands* e disabilitati e si disabilita anche la barra del menu prima di aprire la dialog box, che verrà poi chiusa dal metodo Finish; i comandi e il menu verranno riabilitati dal metodo *Valid* della dialog box *TAbortDialog*.

Sarà opportuno chiarire che le variabili *MaxX* e *MaxY*, inizializzate mediante i valori contenuti nel file di definizione, vengono utilizzate per evitare di stampare righe troppo lunghe o troppe righe in una pagina. In sede di impostazione, quindi, vanno indicati valori corrispon-

Figura 3
Inizio e fine di una fase di stampa.

```
constructor TMyWindow.Init(AParent: PWindowsObject; ATitle: PChar);
begin
  TMyWindow.Init(AParent, ATitle);
  Printer := New(PPrinter, Init);
end;

procedure TMyWindow.FilePrint(var Msg: TMessage);
var
  P: PPrintOut;
begin
  P := New(PWindowPrintout, Init(FileName, @Self));
  Printer^.Print(@Self, P);
  Dispose(P, Done);
end;

procedure TMyWindow.PrinterSetup(var Msg: TMessage);
begin
  Printer^.Setup(@Self);
end;
```

Figura 1 - La semplicità delle fasi di stampa e di impostazione della stampante nel nuovo ObjectWindows *del Borland Pascal 7.0.*

```
constructor TPrinter.Init;
var
  DefaultPrinter: String;
begin
  TObject.Init;    (* azzera tutte le variabili d'istanza *)
  DefaultPrinter := GetEnv('TVPRN');
  if DefaultPrinter <> '' then begin
    if ChangeToTVPRNDIR then
      if not ReadPrinterInfo(DefaultPrinter+'.PRN', PD) then
        PD.Name := '';
    RestoreDirectory;
  end;
  Font := 0;
  TabWidth := 0.5;
  LineSpacing := 1.0 / 6.0;   (* 6 righe per pollice *)
  OkToPrint := False;
end;

destructor TPrinter.Done;
begin
  if PD.Setup.FontList <> nil then
    Dispose(PD.Setup.FontList, Done);
  TObject.Done;
end;
```

Figura 2 - Creazione e distruzione di un'istanza di TPrinter.

```
procedure TPrinter.Start;
var
  R: TRect;
  D: PDialog;
  FileName: String[64];
begin
  if Pos(':', PD.Setup.OutPort) <> 0 then begin (* stampa su stampante*)
    Assign(Lst, Copy(PD.Setup.OutPort, 1, 4));
    (*$I-*) Rewrite(Lst); (*$I+*);
    OkToPrint := not IOError;
  end
  else begin                                    (* stampa su file *)
    R.Assign(2,2,57,9);
    D := New(PPrintFileDialog, Init(R));
    if DeskTop^.ExecView(D) = cmOk then begin
      D^.GetData(FileName);
      Assign(Lst, FileName);
      (*$I-*) Rewrite(Lst); (*$I+*)
      if IOResult <> 0 then begin
        MessageBox(#3'Errore aperture file'#13#3+FileName, nil,
          mfOkButton or mfError);
        OkToPrint := False
      end
      else
        OkToPrint := True;
    end;
    Dispose(D, Done);
  end;
  if OkToPrint then begin
    UserAbort := False;
    Reset;
    BoldFace(False);
    Underline(False);
    SetFont(Font);
  end;
  if OkToPrint then begin
    MaxX := PD.Setup.PaperWidth;
    MaxY := PD.Setup.PaperHeight;
    if PD.Setup.MUnit = 1 then begin      (* misure in centimetri *)
      MaxX := MaxX / 2.54;
      MaxY := MaxY / 2.54;
    end;
    PosX := 0.0;
    PosY := 0.0;
    R.Assign(2,2,31,7);
    DeskTop^.GetCommands(SaveCommands);
    DeskTop^.DisableCommands(SaveCommands);
    MenuBar^.SetState(sfDisabled, True);
    AbortDialog := New(PAbortDialog, Init(R, 'Stampa in corso'));
    Desktop^.Insert(AbortDialog);
  end
  else
    AbortDialog := nil;
end;
  procedure TPrinter.Reset;
  begin
    if OkToPrint and (not UserAbort) then begin
      (*$I-*) Write(Lst, PD.Reset); (*$I+*)
      OkToPrint := not IOError;
    end;
  end;

  procedure TPrinter.Finish;
  begin
    if AbortDialog <> nil then
      AbortDialog^.Close;
    OkToPrint := False;
    (*$I-*) Close(Lst); (*$I+*)
    if IOResult <> 0 then
      ;
  end;
```

denti all'effettiva area di stampa, non alle dimensioni fisiche del foglio di carta. Una stampante HP LaserJet IIIP, ad esempio, quando stampa 10 caratteri per pollice con un'interlinea di 6 righe per pollice su carta A4, non va oltre le 65 righe per pagina con un massimo di 78 caratteri per ognuna.

Impostazione delle caratteristiche di stampa

Il metodo *DoSetup* presiede alla selezione della stampante da utilizzare e alla impostazione di quelle che abbiamo chiamato caratteristiche temporanee (l'elenco dei tipi di carattere disponibili, il formato della carta, il dispositivo di output) di questa o di altre stampanti installate. Vi provvede aprendo una dialog box di tipo *TPrSelectDialog*, dopo aver verificato che la funzione *ChangeToTVPRNDIR* abbia operato con successo, rendendo corrente la directory sede dei file di definizione; in caso contrario, avverte l'utente con un messaggio d'errore.

```
procedure TPrinter.DoSetup;
var
  D: PPrSelectDialog;
  R: TRect;
begin
  if not ChangeToTVPRNDIR then
    MessageBox(#3'Non trovata la directory contenente'#13+
      #3'i file di definizione delle stampanti.',
      nil, mfError or mfOkButton)
  else begin
    R.Assign(1,1,74,13);
    D := New(PPrSelectDialog, Init(R, @Self));
    DeskTop^.ExecView(D);
    Dispose(D, Done);
  end;
  RestoreDirectory;
end;

procedure TPrinter.SelectFont;
var
  D: PFontSelectDialog;
  R: TRect;
begin
  R.Assign(1,1,79,11);
  D := New(PFontSelectDialog, Init(R, Font, @Self));
  if DeskTop^.ExecView(D) = cmOk then
    SetFont(D^.FB^.Focused);
  Dispose(D,Done);
end;

procedure TPrinter.SetFont(F: Word);
begin
  if F >= PD.Setup.FontList^.Count then
    F := PD.Setup.FontList^.Count - 1;
  Font := F;
  CharWidth := 1.0 / PFont(PD.Setup.FontList^.At(Font))^.CPI;
  if OkToPrint and (not UserAbort) then begin
    (*$I-*)
    Write(Lst, (PFont(PD.Setup.FontList^.At(Font))^.Ctrl));
    (*$I+*)
    OkToPrint := not IOError;
  end;
end;

procedure TPrinter.SetTab(T: Real);
begin
  TabWidth := T;
end;
procedure TPrinter.BoldFace(On: Boolean);
begin
  if OkToPrint and (not UserAbort) then begin
    BFace := On;
    (*$I-*)
    if On then
      Write(Lst, PD.BoldOn)
    else
      Write(Lst, PD.BoldOff);
    (*$I+*)
    OkToPrint := not IOError;
  end;
end;

procedure TPrinter.Underline(On: Boolean);
begin
  if OkToPrint and (not UserAbort) then begin
    ULine := On;
    (*$I-*)
    if On then
      Write(Lst, PD.ULOn)
    else
      Write(Lst, PD.ULOff);
    (*$I+*)
    OkToPrint := not IOError;
  end;
end;
```

Figura 4
Impostazione delle caratteristiche di stampa.

Mediante i metodi *SelectFont, SetFont, SetTab, BoldFace* e *Underline* è invece possibile impostare le caratteristiche di ogni singola fase di stampa, scegliendo un tipo di carattere o un intervallo di tabulazione, attivando e disattivando grassetto o sottolineato.

Con *SelectFont* e *SetFont* si sceglie il tipo di carattere. *SelectFont* permette di scegliere in modo interattivo *prima* di iniziare la stampa, attraverso una dialog box di tipo *TFontSelectDialog; SetFont* permette di cambiare il tipo di carattere *durante* la stampa, come abbiamo già visto nel demo illustrato sul numero di novembre.

BoldFace e *Underline* vengono usate per attivare o disattivare, rispettivamente, grassetto e sottolineato, secondo il valore del loro parametro boolean. Il valore del parametro viene assegnato ad una variabile d'istanza (rispettivamente *BFace* e *ULine*), in quanto l'esecuzione dell'uno o dell'altro metodo potrebbe non avere altro effetto, qualora, in fase di impostazione, non si fosse indicata una sequenza di controllo da inviare alla stampante. Questo potrebbe essere necessario se la stampante non disponesse di sequenze per il grassetto o per il sottolineato, ma potrebbe anche risultare comunque opportuno in altri casi; in alcune stampanti, ad esempio, un carattere compresso in grassetto viene «espanso» da 17 a 10 caratteri per pollice, con conseguente alterazione dei conteggi svolti dalla unit per tenere costante cognizione della posizione della testina di stampa. Per fortuna, grassetto e sottolineato possono essere agevolmente emulati; vedremo, quindi, che il metodo *PrintChar* individua i casi in cui la sequenza di controllo per il grassetto o il sottolineato non è stata definita, per provvedere a produrre comunque una stampa conforme ai desideri dell'utente e coerente con l'esigenza di calcolare correttamente la posizione della testina di stampa (è ovvio che tale ultima esigenza non verrebbe soddisfatta nel caso di emulazione del sottolineato su una stampante con tipi di carattere proporzionali, in quanto l'ampiezza del carattere di *underscore* è costante; non mi è mai capitato, tuttavia, di vedere una stampante del genere che non fosse anche in grado di produrre sottolineature efficaci).

Per ora ci fermiamo qui. Il mese prossimo vedremo in dettaglio i metodi che gestiscono e controllano l'invio di caratteri al dispositivo di output.

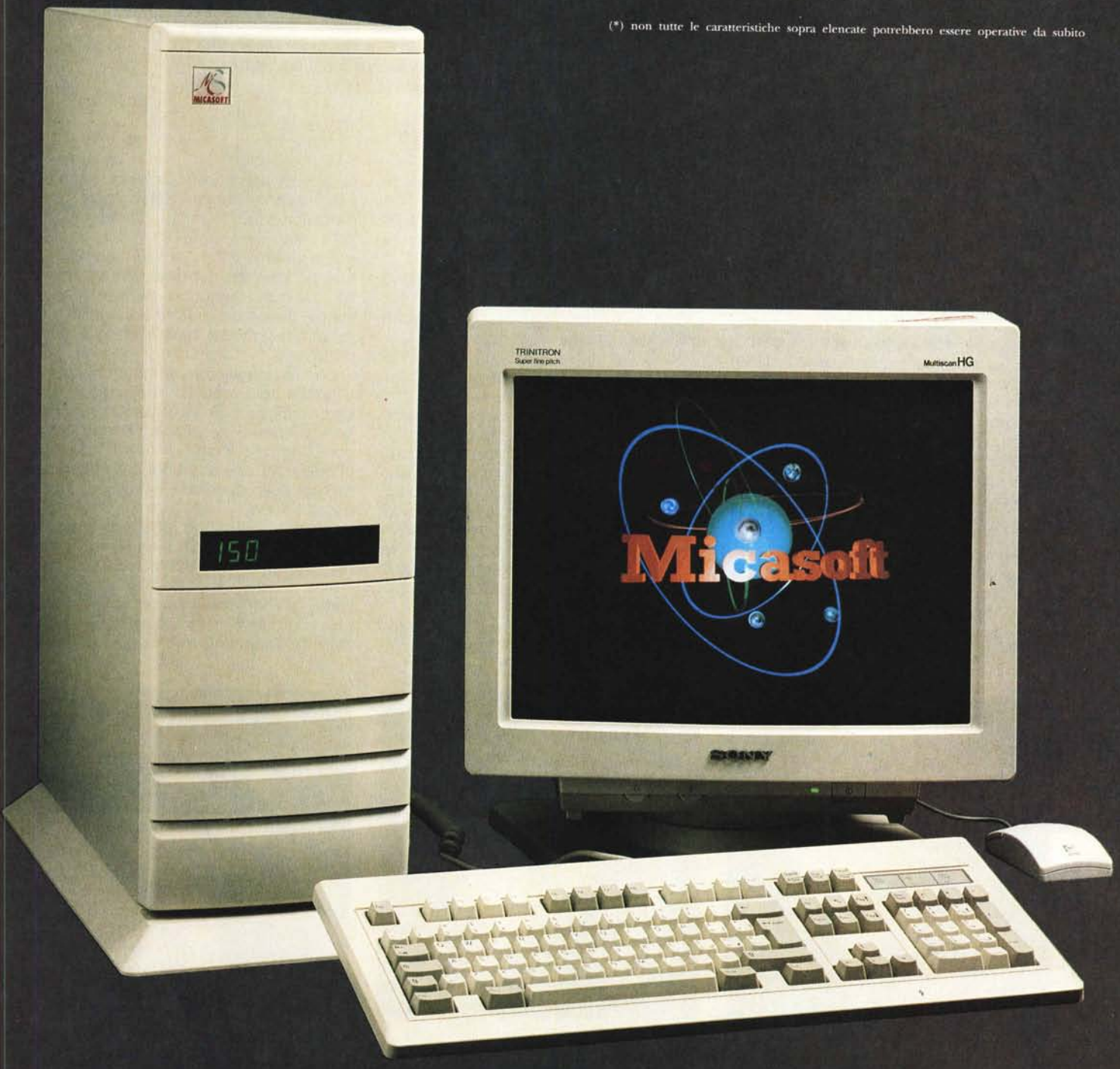
MC

Sergio Polini è raggiungibile tramite MC-link alla casella MC1166 e tramite Internet all'indirizzo MC1166@mclink.it.

MICASOFT esce dalla mischia e propone i suoi nuovi Personal Computers: (*)

componenti elettroniche a norme CEE
test BURN-IN 24h
DR-Dos 6.0 italiano
imballo antiurto che garantisce la consegna integra
garanzia 1 anno su tutte le parti

(*) non tutte le caratteristiche sopra elencate potrebbero essere operative da subito



MICASOFT S.r.l.
Uffici - Via R.R. Pereira, 166 00136 - R O M A
Tell. (06) 3453382 - 3451443 - 3452048 - 348759 - 3497136 FAX (06) 3497295