Under Construction:
Delphi 2 And CGI

by Bob Swart

This time, we're not going to build a component or expert, but we'll start to explore the internet, specifically the world wide web. Our goal is to put a search engine for TDMBKS (*The Delphi Magazine Review Database of Delphi Books*) on the web, including the individual reviews themselves. Our main tool, of course, is Delphi 2...

To write interactive intranet applications, Borland has just released IntraBuilder. In fact, the interactive version of TDMBKS started out as an intranet application at Bolesian. We have an NT Web Server, running Microsoft Internet Information Server (IIS). As "webmaster" of the Delphi pages in our intranet, I wanted to experiment with interactive intranet and internet pages right away. To do this, I had to master HTML and CGI and overcome some problems along the way.

➤ *Above: Figure 1, which shows the result of the HTML code in...*
➤ *Below: Listing 1*

```
<HTML>
<BODY>
<H1>
Are you looking for a good Delphi book?
</H1>
<P>
<HR>
<ADDRESS>
<A HREF="http://home.pi.net/~drbob/">Dr.Bob's Delphi Clinic</A>
</ADDRESS>
</BODY>
</HTML>
```

HTML

HTML stands for HyperText Markup Language, which is the basic language of any static page on the web. Browsers such as Netscape Navigator and Microsoft Internet Explorer are just interpreters of HTML codes in the pages they show. Using special HTML codes, you can do just about anything with your home page, including things like headers, bold and italic text attributes, images and even frames and tables.

I won't go into detail about HTML, I'll just say that if you want to write great looking web pages, then look for a good book to learn it, such as *Netscape & HTML Explorer* from The Coriolis Group, or *(Special Edition) Using HTML* from QUE, and don't depend on HTML editors that (try to) do everything for you automatically. Besides books, one of the best ways to quickly learn HTML is to jump on the web, browse through some nice pages and look in your cache directory (both Navigator and Internet Explorer will save pages in a cache directory to speed up reloading). This is especially useful for special effects such as frames or the use of JavaScript.

The HTML language itself is not hard to learn and you'll be better off if you know what you're writing. I always write my HTML pages in a plain ASCII editor. HTML is made up of plain ASCII text and tags between "<" and ">" characters. An HTML page always starts with <HTML> and ends with </HTML>. The actual contents are put between <BODY> and </BODY> tags. A simple HTML page with a one line header is shown in Listing 1.

The <P> tag denotes a new paragraph, the <HR> is a horizontal rule and between the <ADDRESS> and </ADDRESS> tags we can put address information and a link to a home page or email address, for example. This information will be printed in italic by default.

The <A> tags are part of the foundation of HTML; these form the syntax for the hyper-links, in this case to my own home page at http://home.pi.net/~drbob/. For this simple HTML page, any web browser will show one page with a title and a link (often underlined), as shown in Figure 1.

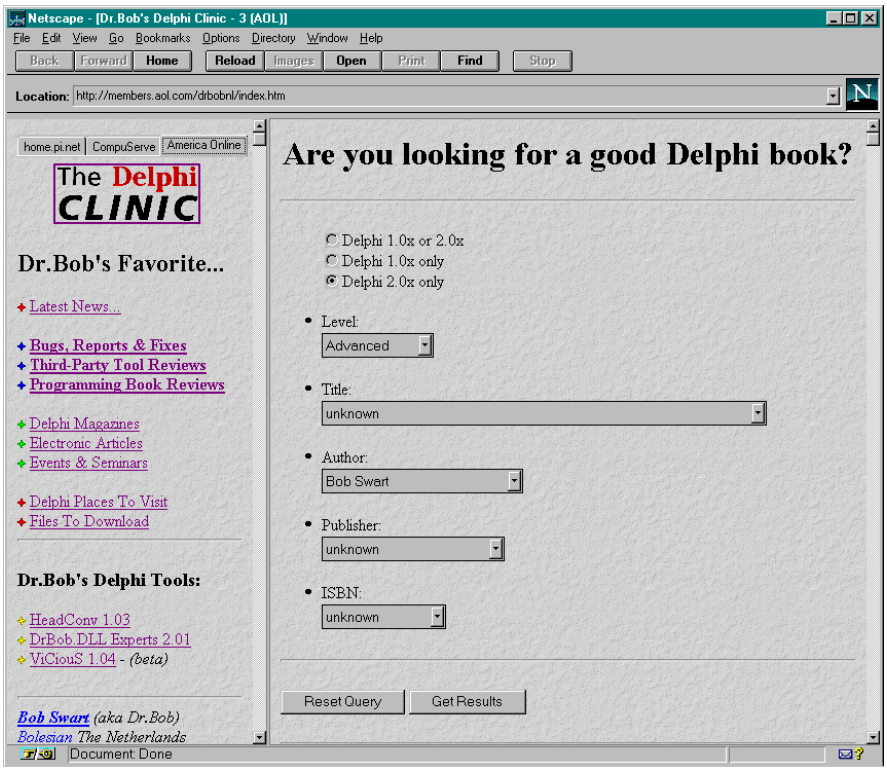
CGI

Whilst HTML is the standard for the hypertext document format, CGI stands for Common Gateway Interface and is the standard communication interface between the Client (Web Browser) and the Server (Web Server). There are at

least two different forms of CGI: the plain 'low level' CGI and a higher level called WinCGI (for Windows (NT) CGI). The first uses environment variables and the standard input and output files, the latter a Windows INI-format file (that specifies the names for the input and output files) to communicate between the Client at the Web Browser and the Web Server running the CGI application.

Delphi 2 CGI applications are non-visual applications, ie `CONSOLE` applications, where the input contains the information (request) sent by the client and the output is the dynamic HTML document that is generated on the fly (and sent back to the Client with the Web Browser). The information that is entered by the Client, and sent to the CGI application to be used when generating the HTML page, can be sent in two ways: either using environment variables and the standard input file (standard CGI) or by using a Windows INI-file and a specified filename (WinCGI).

This article will focus on standard CGI applications. Steve Troxell is working on a follow-up article on the web version of the TDMAid Delphi Magazine Article Index Database application using WinCGI. Both applications will be available for use from The Delphi Magazine's web site of course. Thanks also to TurboPower Software who are kindly hosting these CGI applications on their NT Web Server for us.



➤ *Figure 2*

```
<FORM ACTION="http://www.bolesian.com/scripts/delbooks.exe" METHOD="POST">
<UL>
<INPUT TYPE="radio" NAME="DELPHI" VALUE="0" CHECKED>Delphi 1.0x or 2.0x<BR>
<INPUT TYPE="radio" NAME="DELPHI" VALUE="1">Delphi 1.0x only<BR>
<INPUT TYPE="radio" NAME="DELPHI" VALUE="2">Delphi 2.0x only
<P>
<LI>Level:
<BR><SELECT NAME="Level">
    <OPTION VALUE=""> don't care
    <OPTION VALUE="1"> Beginning
    <OPTION VALUE="2"> Intermediate
    <OPTION VALUE="3"> Advanced
    </SELECT>
<P>
</UL>
<HR>
<P>
<INPUT TYPE="RESET" VALUE="Reset Query">
<INPUT TYPE="SUBMIT" VALUE="Get Results">
</FORM>
```

➤ *Listing 2*

Forms

But before we can determine what the Client wants, let's first take a look at how the Client side actually works. How can a static HTML page cause information to be sent over to the Web Server and an CGI application at the Server to be executed?

For this, we need to use a special HTML extension called *forms*. Just as in Delphi, a form is a place where Windows Controls (such as an edit box, listbox, combobox, button or multi-line text field) can be used. Unlike Delphi, at the moment we have to design our forms non-visually by writing HTML code. Let's take a look at a part of the DELBOOKS.HTM file that is used for this month's application, in Listing 2. For the complete file, please refer to the latest version online at:

`http://members.aol.com/drbobnl/delbooks.htm`.

The code in Listing 2 displays three radio buttons (with choices of `Delphi 1.0x or 2.0x`, `Delphi 1.0x only` and `Delphi 2.0x only`) and a combobox with four items (`Don't care`, `Beginning`, `Intermediate` and `Advanced`). There are also two buttons on the form, one of type `RESET`, to reset the information we've just entered, and one of type `SUBMIT`, to submit the information we've just entered. So, a user with this page loaded in his/her favourite Web Browser just clicks on the `SUBMIT` button, in this case with the text `Get Results` on it.

But how does the Web Server know which CGI application to start for the data that is sent over? For that we need to take a look at the `ACTION` parameter of the `FORM` itself (the first line of Listing 2). The `ACTION` specifies the exact location of the CGI program, in this case it is `http://www.bolesian.com/scripts/delbooks.exe` (but kids, don't try this at home, because this example

is a link to my *intra*net, not the *inter*net).

Of course, the "official" DELBOOKS.HTM contains even more controls (comboboxes, actually) to contain all possible titles, authors, publishers and ISBN numbers of the Delphi books that are in the TDMBKS review database of Delphi books (currently over 40 books). Figure 2 shows this enhanced form in action.

Clicking the `Get Result` button sends the information to the Web Server which will start the DELBOOKS.EXE application (from the form's `ACTION` information) with the information that was supplied in the form. In this case that would be `DELPHI="2", LEVEL="3", TITLE="", AUTHOR="Bob_Swart", PUBLISHER=""` and `ISBN=""`. Note that spaces are replaced by underscores, as I found out the hard way...

The DELBOOKS.EXE Delphi 2 CGI application needs to process the information passed to it, perform the query and generate a dynamic HTML page on the standard output device. The Web Server will then pass this dynamically generated HTML page back to the Web Browser which will show it as the resulting page to the user.

Environment

Standard CGI applications need to look at environment variables to know what kind of communication is used and how big the data in the standard input is. To obtain a list of environment variables with their value, I always use a simple component I wrote some time ago and now compiles with both Delphi 1 and 2, as shown in Listing 3.

This component gets the list of environment variables when it is created. Note the `DosEnvCount` and `DosEnvList` properties are read only, so you need to create this component on the fly (and not drop it on a form), because only then will a 'fresh' list of environment variables be obtained (rather than loaded from the .DFM file).

Parsing

The environment variables the CGI application receives contain a section called `REQUEST_METHOD`. This should be `POST` in our example (I won't go into the other options). Then we need to find the length of the information that was passed to us. For that, we need to look for the `CONTENT_LENGTH` environment variable. The actual information itself is passed as a standard input file of length `CONTENT_LENGTH` (without an end of file marker, so we need to be sure not to read more than the specified number of characters). The data in the standard input file consists of `FIELD=VALUE` pairs, separated by `&` tokens. An example is `AUTHOR="Bob_Swart"&`. Once we have the complete input file in a long string buffer called `Data`, we can quickly scan for the value of `AUTHOR` by using the function shown in Listing 4.

A similar function can be written if the data is numerical (for example for the `LEVEL` field). The skeleton code fragment in Listing 5 shows how to dynamically create the `TBDosEnvironment` variable, read the information from the input file and get ready to parse the values for the controls on the form.

Note the three special `WriteLn` lines that need to be written to the standard output to tell the Web Browser that the dynamically generated page it received is actually a correct HTTP page with `TEXT/HTML` contents.

Database

When writing data-driven CGI applications, you need some way to access your data. One solution is simply to use the BDE and put your data in Paradox or dBase tables. However, if, for some reason, the BDE is not installed on your NT

➤ *Listing 3*

```
unit TBDosEnv;
interface
uses SysUtils, WinTypes, WinProcs, Classes;
type
  TBDosEnvironment = class(TComponent)
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  private
    FDosEnvList: TStringList;
  protected
    function GetDosEnvCount: Word;
  public
    function GetDosEnvStr(Const Name: String): String;
    { This function is a modified version of the GetEnvVar
function that appears
      in the WinDos unit that comes with Delphi; its
interface uses Pascal
      strings instead of null-terminated strings }
    property DosEnvCount: Word read GetDosEnvCount;
    property DosEnvList: TStringList read FDosEnvList;
  end;
implementation
constructor TBDosEnvironment.Create(AOwner: TComponent);
var P: PChar;
    i: Integer;
begin
  inherited Create(AOwner);
  FDosEnvList := TStringList.Create;
 {$IFDEF WIN32}
  P := GetEnvironmentStrings;
 {$ELSE}
  P := GetDosEnvironment; { Win API }
 {$ENDIF}
  i := 0;
  while P^ <> #0 do begin
    Inc(i);
    FDosEnvList.Add(StrPas(P));
    Inc(P, StrLen(P)+1) { Fast Jump to Next Var }
  end;
end {Create};
destructor TBDosEnvironment.Destroy;
begin
  FDosEnvList.Free;
  FDosEnvList := nil;
  inherited Destroy
end {Destroy};
function TBDosEnvironment.GetDosEnvCount: Word;
  { Returns the number of environment variables }
begin
  Result := 0;
  if Assigned(FDosEnvList) then Result := FDosEnvList.Count
end {GetDosEnvCount};
function TBDosEnvironment.GetDosEnvStr(Const Name: String):
String;
var i: Integer;
    Tmp: String;
begin
  i := 0;
  Result := '';
  if Assigned(FDosEnvList) then while i < FDosEnvList.Count
do begin
    Tmp := FDosEnvList[i];
    Inc(i);
    if Pos(Name,Tmp) = 1 then begin
      Delete(Tmp,1,Length(Name));
      if Tmp[1] = '=' then begin
        Delete(Tmp,1,1);
        Result := Tmp;
        i := FDosEnvList.Count { end while-loop }
      end
    end
  end
end {GetDosEnvStr};
end.
```

Web Server (maybe because your friendly neighbourhood Internet Service Provider doesn't provide you with the BDE in the first place), you need to improvise. And since the BDE wasn't installed on our company's intranet Server either (until we installed IntraBuilder, that is), I had to improvise early.

In the old days of Borland Pascal, when we needed data in some kind of database we used a `file of record` type. And that's what we can use when we don't have access to the BDE. All we need to do is write a program that will analyse a table structure, define a `TRecord` type and convert the data from the database to a file of `TRecord`.

Conversion

If we take a look at the general Paradox field types, it shouldn't be hard to notice at least a few which might pose a problem when converting them to Pascal. A `Memo`, for example, usually doesn't fit in a `ShortString`. And what about a `Blob`?

For those types which can be converted, a translation table is shown in Table 1, which at least makes sure no information is lost during the one-way conversion. Using this table, it's not hard to write a simple program that takes a database table as input and generates an ObjectPascal record type as output. See Listing 6.

Records

Our next task is to generate a record type for the DELBOOKS.DB table and convert the actual records to a file of records. Using the RECORD.EXE program (Listing 6), the record format shown in Listing 7 was generated.

Now, all we need to do is write a loop to walk through the database and extract the data from each record in the table, put it in a `TRecord` and write it to a `file of TRecord`. See Listing 8.

A Delphi 2 CGI program can simply open the `file of TRecord` and read the individual records without having to use the BDE. Of course, updating records is not easy, but for that we use the original database (and run the conversion program once we've done an update). After all, I only add new reviews once in a while to the database, so this inconvenience won't matter too much.

Paradox Field Type	ObjectPascal Conversion Type
TStringField (size)	String[length]
TIntegerField, TWordField, TSmallIntField	Integer
Currency	Double
Memo, Blob	Not applicable (ignored)

➤ *Table 1: Paradox to Pascal field type translation*

```
var Data: String;
...
function Value(Const Field: ShortString): ShortString;
var i: Integer;
begin
  Result := '';
  i := Pos(Field+'=',Data);
  if i > 0 then begin
    Inc(i,Length(Field)+1);
    while Data[i] <> '&' do begin
      Result := Result + Data[i];
      Inc(i)
    end
  end
end {Value};
```

➤ *Listing 4*

```
{$APPTYPE CONSOLE}
var Data: String;
    ContentLength,i,j: Integer;
begin
  writeln('HTTP/1.0 200 OK');
  writeln('SERVER: Bolesian Intranet WebServer 1.0');
  writeln('CONTENT-TYPE: TEXT/HTML');
  writeln;
  writeln('<HTML>');
  writeln('<BODY>');
  writeln('<I>Generated by Dr.Bob''s CGI-Expert on </I>',
    DateTimeToStr(Now));
  with TBDosEnvironment.Create(nil) do begin
    for i := 0 to Pred(DosEnvCount) do begin
      if Pos('REQUEST_METHOD',DosEnvList[i]) > 0 then begin
        Data := DosEnvList[i];
        Delete(Data,1,Pos('=',Data))
      end
    end;
    if Data = 'POST' then begin
      ContentLength := StrToInt(GetDosEnvStr('CONTENT_LENGTH'));
      SetLength(Data,ContentLength+1);
      j := 0;
      for i:=1 to ContentLength do begin
        Inc(j);
        read(Data[j]);
      end;
      Data[j+1] := '&';
      { now call Value or ValueAsInteger to obtain individual values }
    end;
  end;
end;
```

➤ *Listing 5*

Performance

One more difference between a CGI application that is able to use the BDE to obtain data and perform queries and our BDE-less CGI application is performance. Not only is our application only about 70Kb in size, it doesn't need the BDE to be loaded, so the load time is much less (resulting in a much faster performance, especially since the complete time from start-up to shut-down counts). Larger scale practical CGI applications using the BDE often use an ISAPI (Information Server API) or NSAPI (Netscape Server API) extension to keep the CGI application "in the air" at all times. I've yet to experiment with these techniques.

In fact, I managed to increase performance even further by not using a `file of TRecord`, but by using an array of records with pre-initialised values! Instead of writing

```
{$APPTYPE CONSOLE}
uses DB, DBTables;
var i: Integer;
begin
  if ParamCount >= 1 then with TTable.Create(nil) do
  try
    TableName := ParamStr(1);
    Active := True;
    writeln('Type');
    writeln('  TRecord = record');
    for i:=0 to Pred(FieldDefs.Count) do begin
      if (FieldDefs[i].FieldClass = TStringField) then
        writeln(' ':4,FieldDefs[i].Name,': String[',FieldDefs[i].Size,'];')
      else begin
        if (FieldDefs[i].FieldClass = TIntegerField) or
          (FieldDefs[i].FieldClass = TWordField) or
          (FieldDefs[i].FieldClass = TSmallintField) then
          writeln(' ':4,FieldDefs[i].Name,': Integer;')
        else
          if (FieldDefs[i].FieldClass = TCurrencyField) then
            writeln(' ':4,FieldDefs[i].Name,': Double;')
          else
            writeln('{ ':6,FieldDefs[i].Name,' }')
      end
    end
  finally
    writeln('  end;');
    Free
  end
  else
    writeln('Usage: record tablename')
end.
```

➤ *Listing 6*

```
{$APPTYPE CONSOLE}
uses DB, DBTables, SysUtils;
var i: Integer;
    Rec: TRecord; { Listing 7 }
    F: File of TRecord;
begin
  if ParamCount >= 1 then with TTable.Create(nil) do
  try
    System.Assign(f,ChangeFileExt(ParamStr(1),'.REC'));
    Rewrite(f);
    TableName := ParamStr(1);
    Active := True;
    First;
    while not Eof do with Rec do begin
      ISBN := FieldByName('ISBN').AsString;
      Title := FieldByName('Title').AsString;
      Author := FieldByName('Author').AsString;
      Publisher := FieldByName('Publisher').AsString;
      Price := FieldByName('Price').AsFloat;
      Code := FieldByName('Code').AsString;
      Level := FieldByName('Level').AsInteger;
      TechnicalContentsQuality :=
         FieldByName('TechnicalContentsQuality').AsInteger;
      QualityOfWriting := FieldByName('QualityOfWriting').AsInteger;
      ValueForMoney := FieldByName('ValueForMoney').AsInteger;
      OverallAssessment := FieldByName('OverallAssessment').AsInteger;
      write(f,Rec);
      Next
    end
  finally
    System.Close(f);
    Free
  end
  else
    writeln('Usage: convert tablename')
end.
```

➤ *Listing 8*

to a file, I generated ObjectPascal code again with the values of the fields. That way, I could generate ObjectPascal source code for a CGI program with all information embedded. No file was needed and after compilation this was a standalone Delphi 2 executable (of 77824 bytes) that contained information on 44 books, was capable of parsing environment variables, reading the standard input file and generating HTML pages on the standard output with dynamic contents depending on the query information entered in the form.

Scoring

The code used to score hits is pretty simple: for fields on the form for which a value was selected, we go through each record in the list of records and add 1 to the score for that particular record if the corresponding field in the record has a value that is equal to (or a

```
Type
  TRecord = record
    ISBN: String[16];
    Title: String[64];
    Author: String[64];
    Publisher: String[32];
    Price: Double;
    Code: String[7];
    { Comments }
    Level: Integer;
    TechnicalContentsQuality:
      Integer;
    QualityOfWriting: Integer;
    ValueForMoney: Integer;
    OverallAssessment: Integer;
    { Cover }
  end;
```

➤ *Listing 7*

sub-string of) the field from the input form. The code for the Author field is as shown in Listing 9.

Note that the `{$IFDEF DEBUG}` can be used to write the value of the input field to the standard output, so we can actually use this CGI application to debug our forms, which can be quite helpful if you think you must have some hits but none show up (because spaces are replaced by underscores, for example). Debugging your CGI applications is otherwise not easy, since you need a Web Server and a Browser to interact for you...

Query Results

Let's take a look at the final part of the CGI application, where we generate the HTML code. I'm using another advanced HTML feature here, namely tables, to format the output in a nice way. For every record that has a score of more than 1, I write the score (number of hits), title, author, publisher, ISBN, level, technical contents, quality of writing, value for money and overall score. I also include a link from the Title to the location of a more detailed review. This is great feature of dynamic HTML pages: you can still include links to static HTML pages, of course, so the result of the query is often the starting point for another set of jumps into hyperspace! See Listing 10.

Once the table header has been written, it's time to go through the individual records. I didn't want to sort them, so for each possible score (from 5 to 1), I just go through the entire list of books and print the one with the current score. That way I know the books will be sorted on the number of

hits, and are still in the order in which they were entered in the original database (which is sorted on level and an overall quality score). So, generally, the books on top of the resulting output list are the best answer to the question that was asked.

The resulting HTML page, generated for the query we saw earlier, is shown in Figure 3. Note that one book had a score of II (for hits on both the Advanced level and the name of the Author). Also note the links from the titles of the books to the more detailed review pages.

➤ *Figure 3*

Generated by Dr.Bob's CGI-Expert on 10/6/96 1:03:41

The following books have been found for you:

Hits	Title	Author	Publisher	ISBN	Level	Con	Wri	Val	Tot
II	The Revolutionary Guide to Delphi 2	Brian Long, Bob Swart, Ewan McNab, Dave Jewell, Arjan Jansen etc	WROX Press	1-874416-67-2	Adv	4	4	3	4
I	Delphi Programming Problem Solver	Neil Rubenking	IDG Books	1-56884-795-5	Adv	5	5	5	5
I	Developing Custom Delphi Components	Ray Konopka	The Coriolis Group	1-883577-47-0	Adv	5	4	4	4
I	Delphi 2 Developer's Guide	Xavier Pacheco & Steve Teixeira	SAMS	0-672-30914-9	Adv	5	5	4	5
I	Peter Norton's Guide to Delphi 2	John Paul Mueller	SAMS Premier	0-672-30898-3	Adv	4	4	4	4
I	Database Developer's Guide with Delphi 2	Ken Henderson	SAMS	0-672-30862-2	Adv	5	5	4	5

Thank you for using the TDMBKS on-line review database of Delphi Books!

Dr.Bob's Delphi Clinic

Conclusion

I hope to have shown that you can write interactive internet and intranet CGI applications using Delphi 2 in a fairly straightforward (albeit non-visual) manner. Personally, I plan to do a lot more with Delphi and the internet/intranet, so stay tuned for more news and technical stuff. The best places to check out for news and more information are of course The Delphi Magazine and my own internet home page at `http://home.pi.net/~drbob/`. Watch out too for a new CGI-Expert for Delphi 2 that I'm working on...

If you have any experience, problems or interesting ideas on internet development using Delphi, don't hesitate to send me a message at bob@bolesian.nl. We're all here to learn from each other. That's the fastest way to master it!

See you next time for the promised `TRuleBase` component.

Bob Swart (aka Dr.Bob) is a full-time Knowledge Engineer Specialist for Bolesian in The Netherlands and part-time technical author and columnist for The Delphi Magazine. In his spare time he likes to watch video tapes of Star Trek Voyager and Deep Space Nine with his 2.5 year old son Erik Mark Pascal.

➤ *Listing 9*

```
if DataRec.Author <> '' then begin
 {$IFDEF DEBUG}
  writeln('Author: ',DataRec.Author,'<BR>');
 {$ENDIF}
  for i:=1 to Books16 do
    if Pos(DataRec.Author,Book16[i].Author) > 0 then Inc(Result16[i]);
    for i:=1 to Books32 do
      if Pos(DataRec.Author,Book32[i].Author) > 0 then Inc(Result32[i])
end;
```

➤ *Listing 10*

```
writeln('<HR>');
writeln('<P>');
writeln(
  '<H3>The following books have been found for you:</H3>');
writeln('<TABLE BORDER>');
writeln('<TR>');
writeln('<TH><B>Hits</B></TH>');
writeln('<TH><B>Title</B></TH>');
writeln('<TH><B>Author</B></TH>');
writeln('<TH><B>Publisher</B></TH>');
writeln('<TH><B>ISBN</B></TH>');
writeln('<TH><B>Level</B></TH>');
writeln('<TH>Con</TH>');
writeln('<TH>Wri</TH>');
writeln('<TH>Val</TH>');
writeln('<TH><B>Tot</B></TH>');
writeln('</TR>');
if DataRec.Delphi2 then begin
  for Hits := 5 downto 1 do begin
    for i:=1 to Books32 do if Result32[i] = Hits then begin
      writeln('<TR>');
      writeln('<TD>',Roman[Hits],'</TD>');
      writeln('<TD><A HREF="',root32,Book32[i].HREF,'">',
        Book32[i].Title,'</A></TD>');
      writeln('<TD>',Book32[i].Author,'</TD>');
      writeln('<TD>',Book32[i].Publisher,'</TD>');
      writeln('<TD>',Book32[i].ISBN,'</TD>');
      writeln('<TD>',Level[Book32[i].Level],'</TD>');
      writeln('<TD>',Book32[i].TechnicalContentsQuality,
        '</TD>');
      writeln('<TD>',Book32[i].QualityOfWriting,'</TD>');
      writeln('<TD>',Book32[i].ValueForMoney,'</TD>');
      writeln('<TD><B>', Book32[i].OverallAssessment,
        '</B></TD>');
      writeln('</TR>')
    end
  end;
  if DataRec.Delphi1 then writeln('<TR></TR>')
end;
if DataRec.Delphi1 then begin
  for Hits := 5 downto 1 do begin
    for i:=1 to Books16 do if Result16[i] = Hits then begin
      writeln('<TR>');
      writeln('<TD>',Roman[Hits],'</TD>');
      writeln('<TD><A HREF="',root16,Book16[i].HREF,'">',
        Book16[i].Title,'</A></TD>');
      writeln('<TD>',Book16[i].Author,'</TD>');
      writeln('<TD>',Book16[i].Publisher,'</TD>');
      writeln('<TD>',Book16[i].ISBN,'</TD>');
      writeln('<TD>',Level[Book16[i].Level],'</TD>');
      writeln('<TD>',Book16[i].TechnicalContentsQuality,
        '</TD>');
      writeln('<TD>',Book16[i].QualityOfWriting,'</TD>');
      writeln('<TD>',Book16[i].ValueForMoney,'</TD>');
      writeln('<TD><B>',Book16[i].OverallAssessment,
        '</B></TD>');
      writeln('</TR>')
    end
  end
end;
writeln('</TABLE>');
writeln('<HR>');
writeln('<A HREF="http://home.pi.net/~drbob/">Dr.Bob''s
Delphi Clinic</A>');
writeln('</BODY>');
writeln('</HTML>');
writeln;
Free;
```