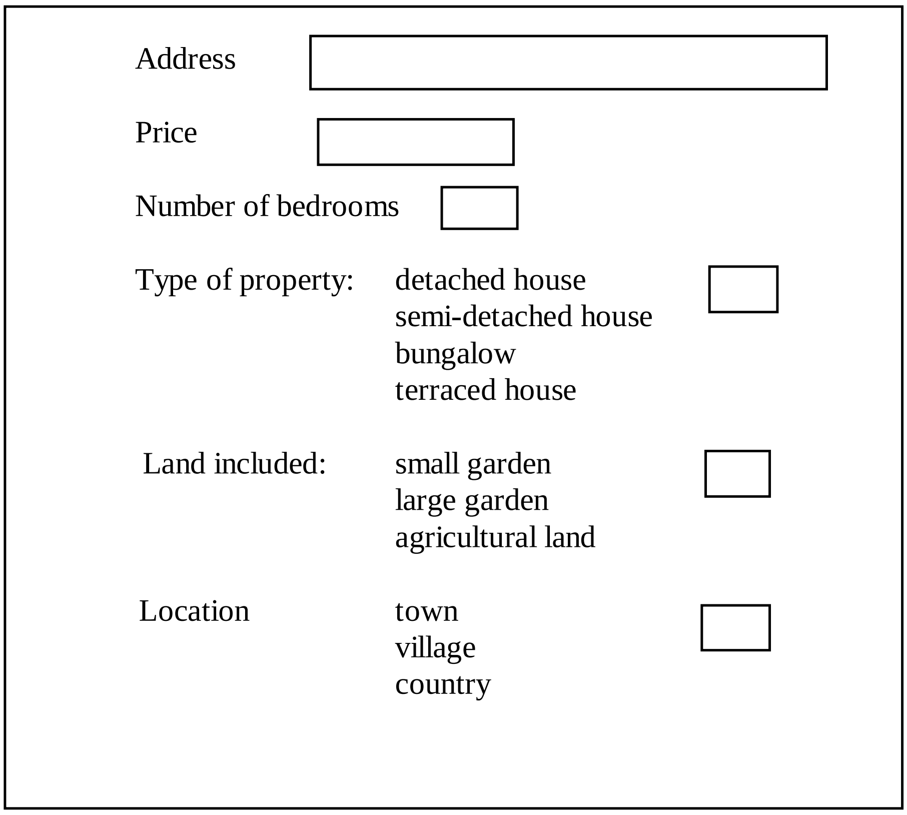
THIRTEEN

Developing a database (1)

This chapter is the first of two in which we look in detail at a database for an estate agent. The program will use many of the ideas introduced previously about saving data on disc, but we will also see how the SEEK command can allow individual records to be accessed in a file, then edited or deleted. We will set up search facilities to pick out particular records from a file, and see how a database system can use multiple files:

Application Estate agent's database

You are asked to set up a program in Delphi which will store details of houses for sale by an estate agent. For each property, the following information is to be recorded:



Address	
Price	
Number of bedrooms	
Type of property:	detached house semi-detached house bungalow terraced house
Land included:	small garden large garden agricultural land
Location	town village country

Once records have been entered, it should be possible to enter customers' requirements and the computer will display details of those properties which are suitable. Questions to be asked of the customer are:

Customer name	
Maximum price	
Minimum number of bedrooms required	
Type of property required (or no preference)	
Amount of land required (or no preference)	
Location required (or no preference)	

The following test data can be used:

HOUSES AVAILABLE

Sea View, Fairbourne	£38000	2 bedrooms
bungalow in village with small garden		
37 High Street, Porthmadog	£42000	3 bedrooms
terraced house in town with small garden		
Pant Mawr Farmhouse, Bala	£164000	4 bedrooms
detached house in country with agricultural land		
The Old Chapel, Tanygrisiau	£58000	4 bedrooms
detached house in village with large garden		
4 Barmouth Road, Dolgellau	£96000	4 bedrooms
semi-detached house in town with large garden		
Tal y Bont Cottage, Borth	£54000	2 bedrooms
detached house in village with small garden		

CUSTOMER REQUIREMENTS

Aled Jenkins	
price not over £60000	minimum 3 bedrooms
Stuart Humphries	
price not over £100000	minimum 2 bedrooms
must be: in village, with large garden	
Ian Andrews	
price not over £200000	minimum 3 bedrooms
must have agricultural land	
Elizabeth Edwards	
price not over £100000	minimum 2 bedrooms
must be a detached house	

Begin the program by setting up a directory ESTATE and saving a Delphi project into it. Use the Object inspector to **Maximize** the Form, and drag the dotted grid to nearly fill the screen.

The project we are undertaking is quite complex, so it would be useful to draw a top down structure diagram to clarify the design. This is done on the next page.

The project can be divided into two main sections:

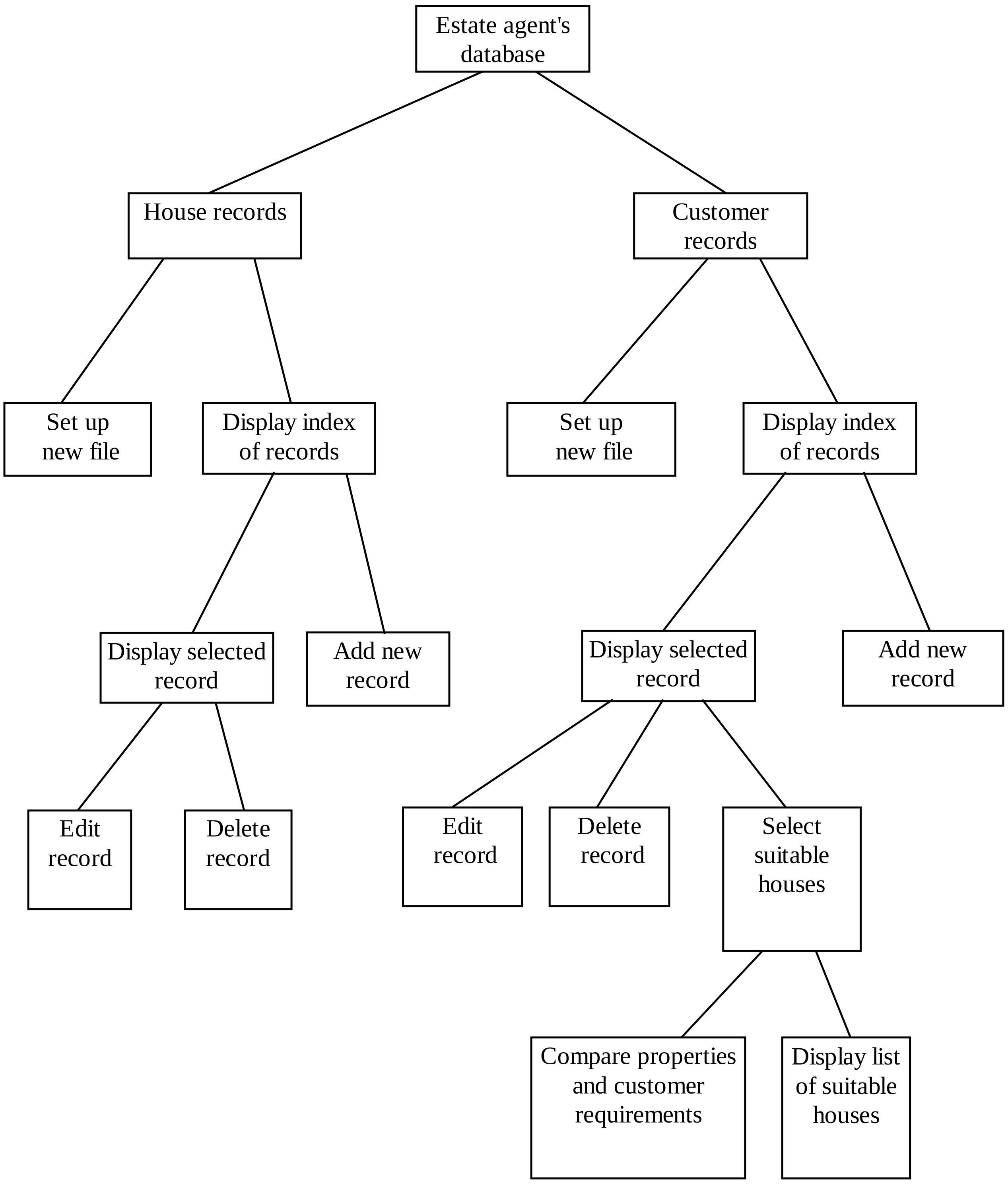
- **house records system** to keep details of the properties available
- **customer records system** to keep details of customers and their requirements, and to select properties which are suitable for each customer.

Within each of these sections there will be a series of data processing operations required:

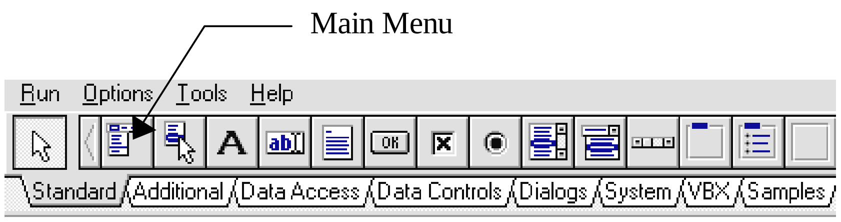
- setting up a **new file** ready to store records
- **adding** a record to the file
- **selecting** and **displaying** a particular record from the file
- **editing** an existing record
- **deleting** a record which is no longer required, for example when a house is sold or a customer purchases a house.

An additional procedure will be needed to check for suitable properties to meet the customers' requirements.

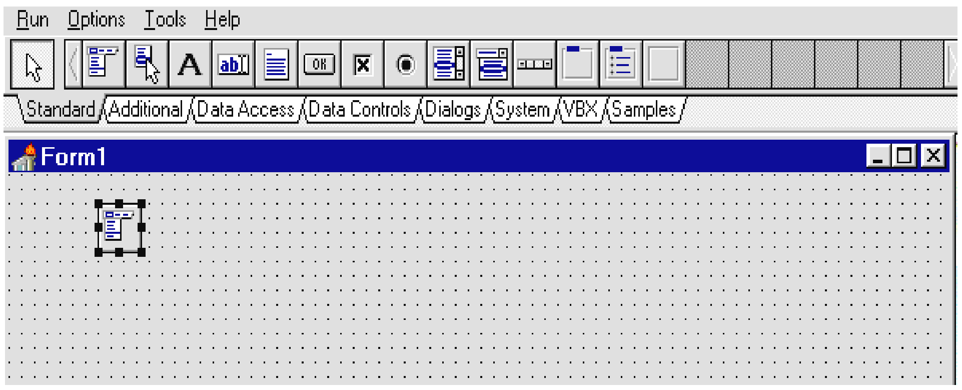
Top down structure diagram



The project is going to require a complex menu system, but fortunately Delphi provides an easy way to set this up. Go to the STANDARD component menu and press the '**Main Menu**' button:

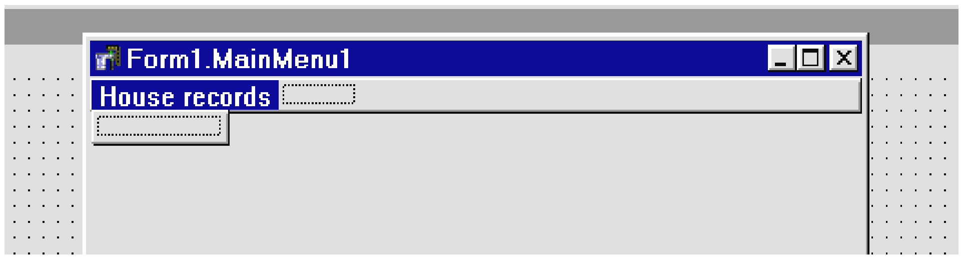

Main Menu
Run Options Tools Help
Standard Additional Data Access Data Controls Dialogs System VBX Samples

Drag the 'Main Menu' component down onto the form:


Run Options Tools Help
Standard Additional Data Access Data Controls Dialogs System VBX Samples
Form1

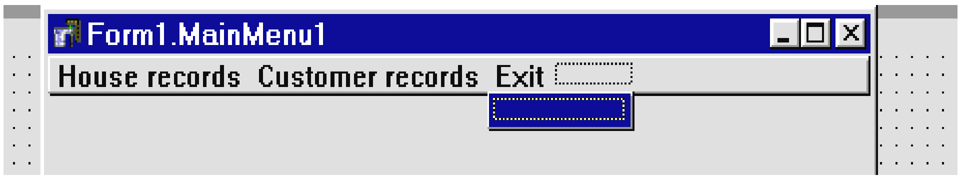
The '**Main Menu**' is different other components we have used. It appears as a fixed size icon and can be placed anywhere on the form - the position does not matter.

Double-click the icon and a new window will appear to show the menu system we are constructing. Press ENTER to bring up the Object Inspector window, then type the caption '**House records**'. Press ENTER and this will appear in the blue box on the menu bar:



Click on the top line alongside the 'House records' caption. The blue box will move to this point. Press ENTER to show the Object Inspector then add a caption '**Customer records**'.

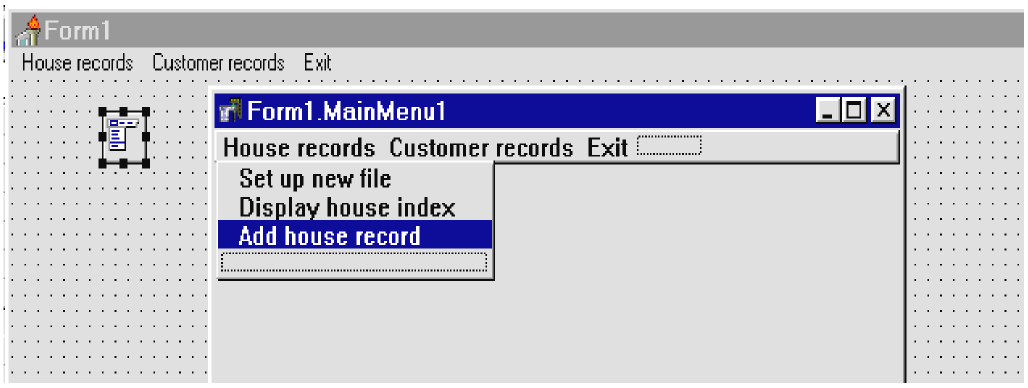
Again click alongside and complete the main menu with an '**Exit**' option:



House records

We will now develop the part of the database which deals with house records.

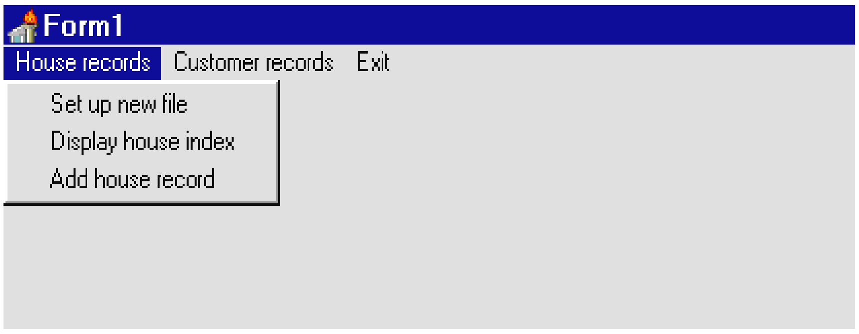
Click on 'House records' and a box will appear underneath. Click the box then enter the caption '**Set up new file**'. Press ENTER and a further box will appear underneath.


Form1
House records Customer records Exit
Form1.MainMenu1
House records Customer records Exit
Set up new file
Display house index
Add house record

Add the remaining menu items in a similar way:

'Display house index'
'Add house record'

Compile and run the program. The three menu options 'House records', 'Customer records' and 'Exit' appear on the top line of the screen. Clicking 'House records' should produce a drop-down menu with the list of choices:



Return to the Delphi editing screen. Bring the **Form1** window to the front. Double-click the word '**Exit**' on the menu line of the Form and an event handler procedure will be created. Add a '**halt**' command to this:

```
procedure TForm1.Exit1Click(Sender: TObject);
begin
  halt;
end;
```

Click on the '**House records**' caption at the top of the grid. The drop-down menu should appear. Click '**Set up new file**' and an event handler procedure appears. Add the lines:

```
procedure TForm1.Setupnewfile1Click(Sender:TObject);
begin
  assignfile(housefile,'houses.dat');
  rewrite(housefile);
  closefile(housefile);
end;
```

Before we can use the program we must define the *fields* for a record to store house data. Go to the **type** heading at the top of the program and add lines just below the heading:

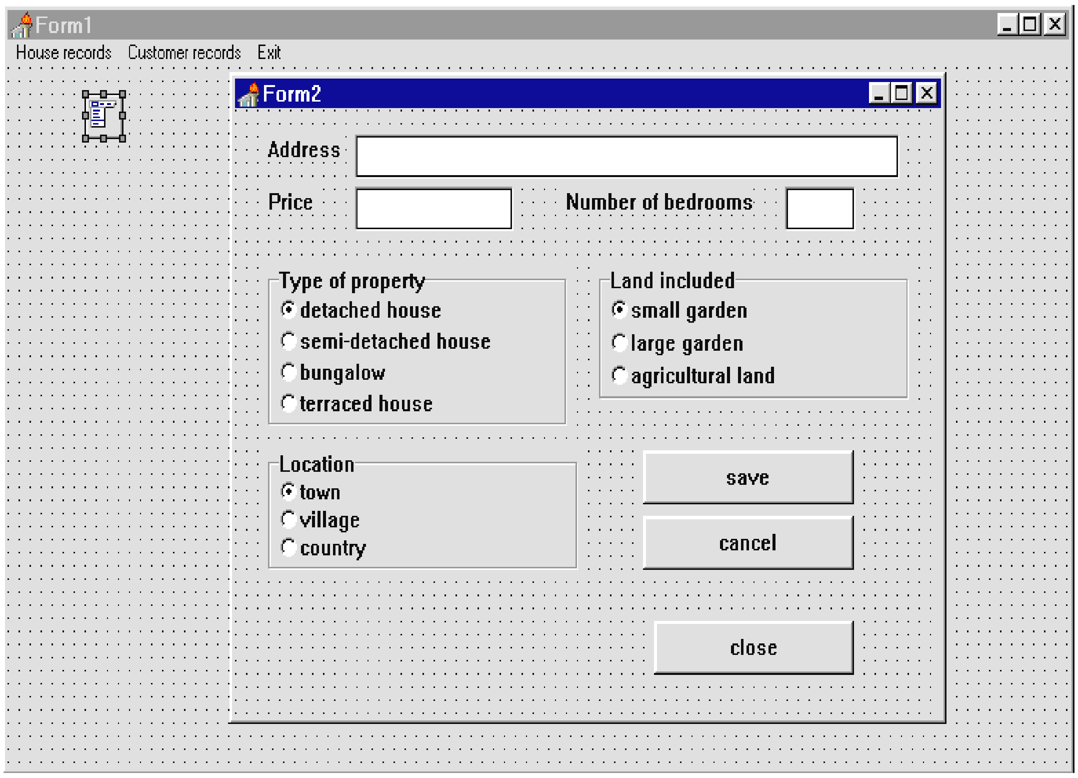
```
type
  house=record
    address:string[50];
    price:real;
    bedrooms:integer;
    housetype,land,location:integer;
  end;
  TForm1=class(TForm)
     .....
```

We also need to add lines to the *Public declarations* to allocate variable names:

```
  public
    { Public declarations }
    houserecord:house;
    housefile:file of house;
  end;
```

The next section we will develop is the '**Add house record**' option.

Go to the Delphi editing screen and click the short-cut button to **add a new form**. Choose '*Blank form*'. Use the Object Inspector to set its **BorderStyle** property to '**Dialog**', and its **FormStyle** property to '**StayOnTop**'.



Set up Form2 for input of house details by adding the components shown:

- Three *Edit Boxes*, and *Labels* with the captions: '**Address**', '**Price**' and '**Number of bedrooms**'.
- A *Radio Group* with the caption '**Type of property**'. Double-click alongside the **Items** property to enter the labels for four buttons: '**detached house**', '**semi-detached house**', '**bungalow**' and '**terraced house**'.
- A *Radio Group* with the caption '**Land included**'. This should contain three buttons with the labels: '**small garden**', '**large garden**', and '**agricultural land**'.
- A *Radio Group* with the caption '**Location**'. This should contain three buttons with the labels: '**town**', '**village**', and '**country**'.
- Three *Buttons* with the captions: '**save**', '**cancel**', and '**close**'.

For each of the *Radio Groups*, set the **ItemIndex** property to **0**.

Use the Project Manager to bring the *Unit1* program window to the front. Add '*unit2*' to the '**uses**' list:

```
uses
  SysUtils,WinTypes,.....Dialogs,Menus,unit2;
```

Go to *Form1* and click '**House records**' on the menu line, followed by '**Add house record**'. An event handler will appear. Add the command to open the *Form2* window:

```
procedure TForm1.AddhouserecordlClick(Sender:TObject);
begin
  form2.visible:=true;
end;
```

Now use the *Project Manager* to bring the Form2 window to the front. Double-click the '**close**' button and add a command to the event handler to close the Form:

```
procedure TForm2.Button3Click(Sender: TObject);
begin
  form2.visible:=false;
end;
```

Build and run the program. Select '**House records/Add house record**'. The house input window *Form2* should appear. Click the '**close**' button and the window should disappear. Click '**exit**' to return to the Delphi editing screen.

Go to the *Unit2* program screen. It would be useful to have a procedure to blank out the Edit Boxes on the house input screen and to reset the *Radio Group* buttons. Insert a procedure to do this at the end of the program:

```
procedure TForm2.clear;
begin
  edit1.text:='';
  edit2.text:='';
  edit3.text:='';
  radiogroup1.itemindex:=0;
  radiogroup2.itemindex:=0;
  radiogroup3.itemindex:=0;
end;
```

Add the '**clear**' procedure to the list under the **type** heading:

```
type
    ....
   Button3: TButton;
   procedure Button3Click(Sender: TObject);
   procedure clear;
      ....
```

The procedure can be used to blank out the Edit Boxes if the user presses '**cancel**'. Double-click the '**cancel**' button and add a line:

```
   procedure TForm2.Button2Click(Sender: TObject);
   begin
     clear;
   end;
```

Also add '**clear**' to the '**close**' button click procedure:

```
   procedure TForm2.Button3Click(Sender: TObject);
   begin
     clear;
     form2.visible:=false;
   end;
```

Build and run the program. Set up a new house file then go to the '**Add house record**' screen. Check that text or figures can be entered in the three Edit Boxes, and that this can be cleared by pressing the '**cancel**' button. Close the house input window with text still showing in the Edit Boxes and check that this has been cleared if you reselect the '**Add house record**' option. Exit from the program and return to the Delphi editing screen.

We can now set up event handlers to transfer data into the fields of the house record. Double-click on the '**Address**' Edit Box and add the line:

```
   procedure TForm2.Edit1Change(Sender: TObject);
   begin
     form1.houserecord.address:=edit1.text;
   end;
```

'**Price**' should be input as a real number. Double-click the '**price**' Edit Box to create the event handler, then add the lines:

```
   procedure TForm2.Edit2Change(Sender: TObject);
   begin
     if edit2.text='' then
       form1.houserecord.price:=0
     else
       form1.houserecord.price:=strtofloat(edit2.text);
    end;
```

The '**Number of bedrooms**' Edit Box requires a similar event handler, but this time the input is an integer. Double-click the Edit Box and add the lines:

```
procedure TForm2.Edit3Change(Sender: TObject);
   begin
     if edit3.text='' then
```

```
      form1.houserecord.bedrooms:=0
    else
      form1.houserecord.bedrooms:=strtoint(edit3.text);
  end;
```

We must finish by telling the computer that *Form2* will be accessing '**houserecord**' which belongs to *Form1*. Do this by adding a '**uses**' entry under the '**implementation**' heading:

```
     implementation
     {$R *.DFM}
     uses
       unit1;
```

Run the program. Set up a new house file then go to the '**Add house record**' screen. Check that the '**price**' and '**bedrooms**' Edit Boxes are error trapped to only accept numbers. Return to the Delphi editing screen.

Double-click the '**save**' button to set up its event handler, and add lines of program to save the record into the house file. The procedure is similar to one we wrote in the *airport* program in chapter 6:

```
procedure TForm2.Button1Click(Sender: TObject);
begin
  with form1 do
  begin
    houserecord.housetype:=radiogroup1.itemindex;
    houserecord.land:=radiogroup2.itemindex;
    houserecord.location:=radiogroup3.itemindex;
    assignfile(housefile,'houses.dat');
    reset(housefile);
    seek(housefile,filesize(housefile));
    write(housefile,houserecord);
    closefile(housefile);
  end;
  clear;
end;
```

Remember that ***houserecord*** and ***housefile*** are variables belonging to *Form1*. The command:

with form1 do...

is telling the computer to assume that variables belong to *Form1*. This saves us having to use the '**form1.**' prefix every time.

For example, we can write:

assignfile(housefile,'houses,.dat');

instead of:

```
assignfile(form1.housefile,'houses.dat');
```

The lines:

```
houserecord.housetype:=radiogroup1.itemindex;
houserecord.land:=radiogroup2.itemindex;
houserecord.location:=radiogroup3.itemindex;
```

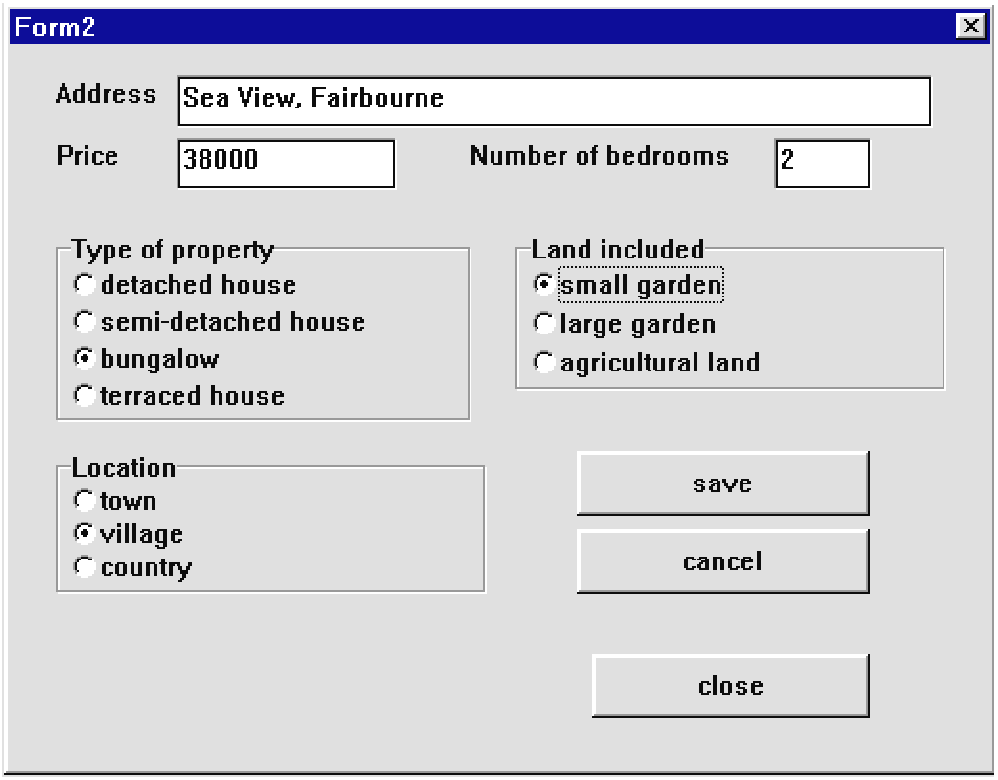
record the numbers of the buttons selected in the *Radio Groups* as codes for: the type of house, land included, and location of the property.

The file is opened using:

```
assignfile(housefile,'houses.dat');
reset(housefile);
```

The SEEK command moves to the end of the existing file and the new record is added. Finally, the file is closed so that the data is secure:

```
seek(housefile,filesize(housefile));
write(housefile,houserecord);
closefile(housefile);
```


Form2

Address	Sea View, Fairbourne
Price	38000
Number of bedrooms	2

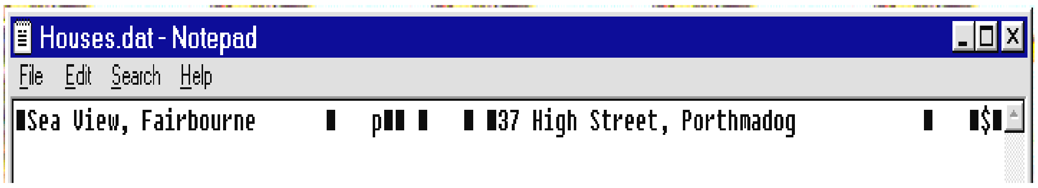
Type of property: detached house / semi-detached house / [✓ bungalow] / terraced house

Land included: [✓ small garden] / large garden / agricultural land

Location: town / [✓ village] / country

We can now test the input procedure:

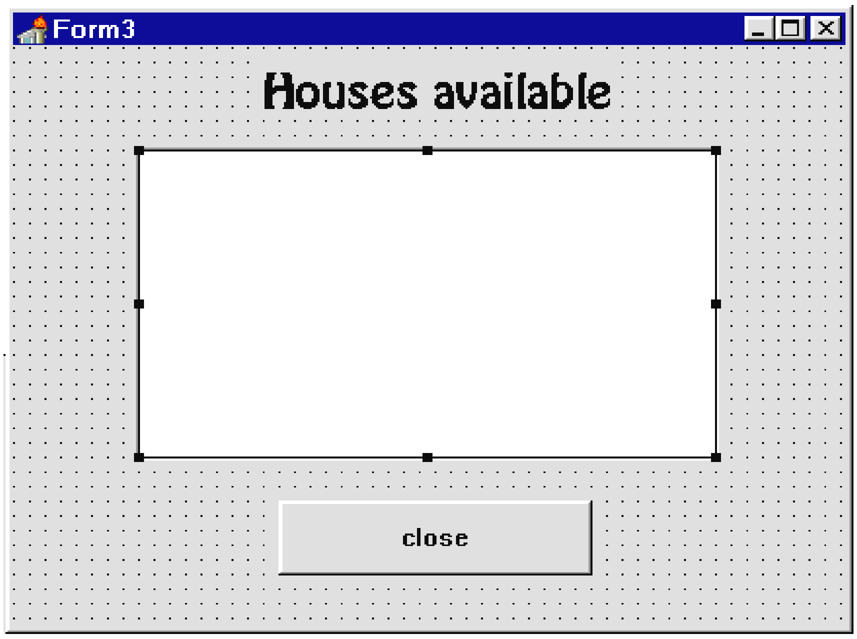
- Build and run the program.
- Choose '**House records/Set up new file**'.
- Choose '**House records/Add house record**' and enter the first house from the list of test data: *Sea View, Fairbourne*. Click the '**save**' button to transfer this to disc.
- Enter the record for the second property, *37 High Street, Porthmadog*, and save this.
- Click ' **close**' to return to *Form1*, then click '**Exit**' to end the program.
- Check that the records have been saved on disc using the NOTEPAD utility. Select the file '**houses.dat**' and the addresses of the two properties should appear:



```
▮Sea View, Fairbourne      ▮   p▮▮ ▮   ▮ ▮37 High Street, Porthmadog          ▮   ▮$▮
```

Don't worry that the **price**, **bedrooms**, and **property codes** are not shown. NOTEPAD only displays text, not data which is in number format.

Return to the Delphi editing screen. Use the '**New form**' short-cut button to create **Form3** which will be used to display a list of the houses. Set the **BorderStyle** property to '**Dialog**' and the **FormStyle** property to '**StayOnTop**'. Add a List Box to the form, along with a *Label* '**Houses available**', and a *Button* with the caption '**close**':



To allow *forms 1* and *3* to exchange data, add **unit3** to the '**uses**' line of *Form1*:

```
       SysUtils, WinTypes,....unit2,unit3;
```

and add a '**uses**' line to the **implementation** section of **Form3**:

```
          implementation
          {$R *.DFM}
          uses
            unit1;
```

Double-click the '**close**' button on **Form3** to create an event handler and add the line:

```
  procedure TForm3.Button1Click(Sender: TObject);
  begin
    form3.visible:=false;
  end;
```

Go to *Form1* and click '**House records/Display house index**'. Add a line to the event handler which is created:

```
  procedure TForm1.Displayhouseindex1Click
                               (Sender: TObject);
  begin
    form3.visible:=true;
  end;
```

Build and run the program. Select the option '**Display house index**'. Check that **Form3** appears, and that it disappears again when the '**close**' button is pressed. Return to the Delphi editing screen.

We can now write a procedure to display the house addresses in the *List Box* on *Form3*. Bring the *Form3* window to the front. Click on the dotted grid and press ENTER to bring up the Object Inspector. Click the **Events** tab, then double click alongside '**OnActivate**'. This creates an event handler which will operate each time the *Form3* window is opened on the screen. Add lines to the procedure:

```
  procedure TForm3.FormActivate(Sender: TObject);
  begin
    listbox1.clear;
    assignfile(form1.housefile,'houses.dat');
    reset(form1.housefile);
    while not eof(form1.housefile) do
    begin
      read(form1.housefile,form1.houserecord);
      listbox1.items.add(form1.houserecord.address);
    end;
  end;
```

This begins by blanking out the list box. We then open the house file:

```
**assignfile(form1.housefile,'houses.dat');**
**reset(form1.housefile);**
```

A loop continues to repeat as long as the end of the file has not yet been reached:

while not eof(form1.housefile) do ...

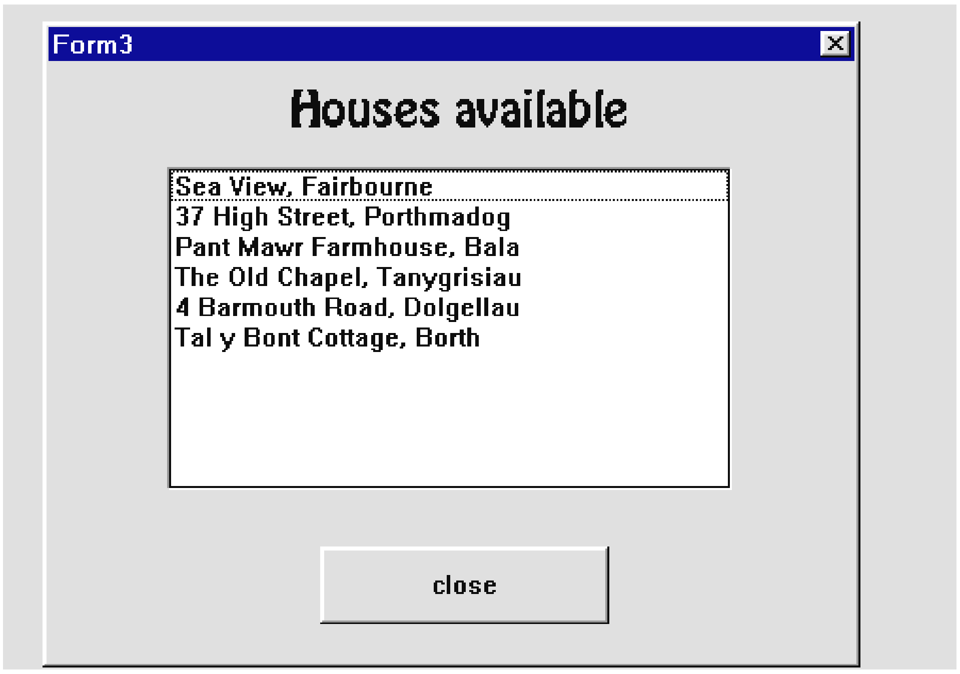
Each time around the loop the next house record is loaded, and the **address** field is copied into the *List Box*:

read(form1.housefile, form1.houserecord);
listbox1.items.add(form1.houserecord.address);

We just need another procedure to close the ***Form3*** window if some other menu option is selected. Bring up the Object Inspector for ***Form3*** again and select the **Events** list. Double-click alongside '**OnDeactivate**' and add the line:

```
procedure TForm3.FormDeactivate(Sender: TObject);
begin
  form3.visible:=false;
end;
```

Build and run the program. Select the '**Display house index**' option. Check that the two houses you entered earlier are listed.



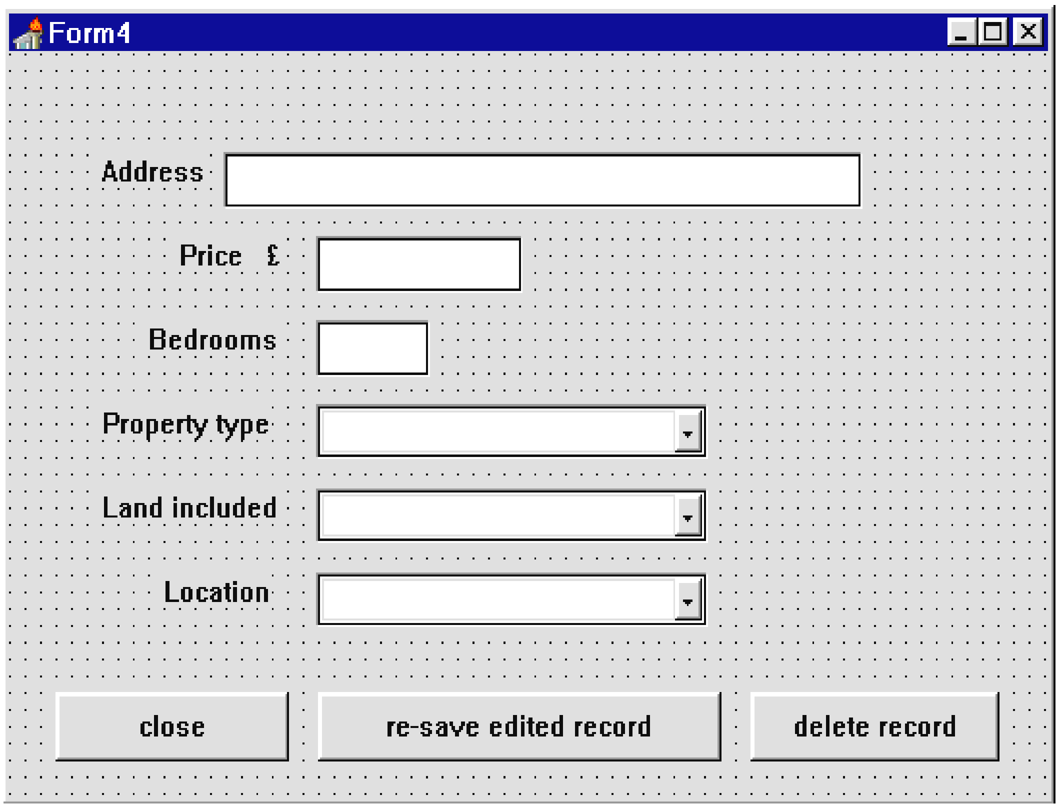
```
Sea View, Fairbourne
37 High Street, Porthmadog
Pant Mawr Farmhouse, Bala
The Old Chapel, Tanygrisiau
4 Barmouth Road, Dolgellau
Tal y Bont Cottage, Borth
```

Go to the '**Add house record**' option and add the remaining four houses in the table of test data on page 244. Return to the house index screen and check that all six properties are now listed. Return to the Delphi editing screen.

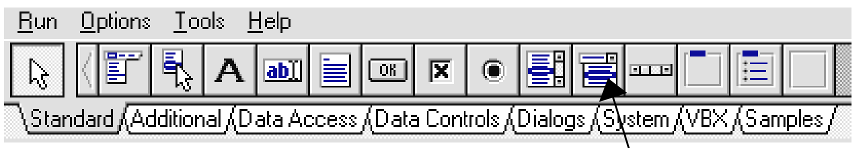
The next step is to work on a display screen for the house records. Use the '**New form**' short-cut button to create **Form4**.

Bring up the Object Inspector and set the **FormStyle** property to '**StayOnTop**' and the **BorderStyle** property to '**Dialog**'.

Add three *Edit Boxes* to **Form4**, with *Labels* alongside for '**Address**', '**Price**' and '**Bedrooms**':

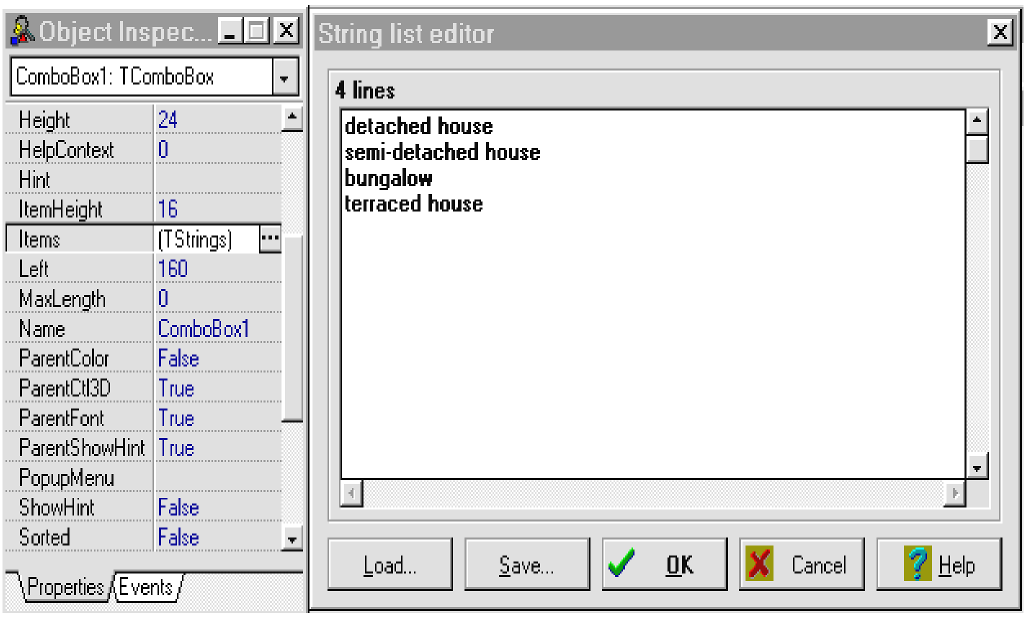


We are going to display the 'Property type' using a **Combo Box** component - this is selected from the STANDARD menu:


Run Options Tools Help
Standard Additional Data Access Data Controls Dialogs System VBX Samples
Combo Box

Use the mouse to position the *Combo Box* on the form in the same way as an Edit Box. The only difference you notice is a small button at the right hand side of the box with a downwards pointing arrow. Place a *Label* alongside, with the caption '**Property type**'.

Add two more *Combo Boxes*, with *Labels* alonside for '**Land included**' and '**Location**'.



ComboBox1: TComboBox

Height	24
HelpContext	0
Hint	
ItemHeight	16
Items	(TStrings)
Left	160
MaxLength	0
Name	ComboBox1
ParentColor	False
ParentCtl3D	True
ParentFont	True
ParentShowHint	True
PopupMenu	
ShowHint	False
Sorted	False

4 lines

detached house
semi-detached house
bungalow
terraced house

Select the 'Property type' *Combo Box* and press ENTER to bring up the Object Inspector. Double-click in the right hand column alongside '**Items**' and the **String List Editor** window will appear. Enter a list of the four types of property:

detached house
semi-deteched house
bungalow
terraced house

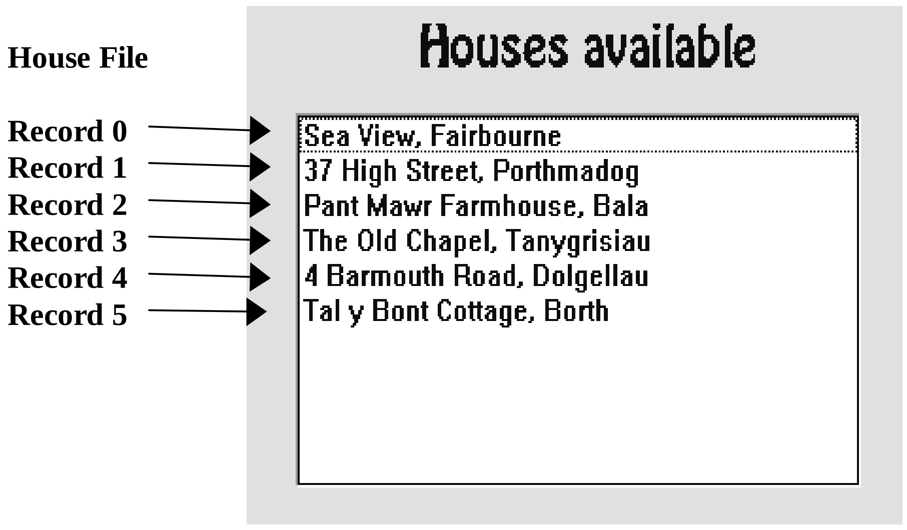
When the program runs, the Combo Box will be given the code number for the type of property. It can then select and display the correct description from this list.

Set up the **'Items'** lists in a similar way for the other two *Combo Boxes*:

Land included:	small garden
	large garden
	agricultural land

Location:	town
	village
	country

Complete **Form4** by adding three Buttons with the captions: '**close**', '**re-save edited record**', and '**delete record**'.
We need **Form4** to appear and display the details of a property when the user clicks on one of the addresses in the '**Houses available**' list. The position of the address in the list corresponds to the position of the record in the file:



Sea View, Fairbourne
37 High Street, Porthmadog
Pant Mawr Farmhouse, Bala
The Old Chapel, Tanygrisiau
4 Barmouth Road, Dolgellau
Tal y Bont Cottage, Borth

Bring the **Form3** window to the front and double-click on the *ListBox* to produce an event handler. Add the lines:

```
procedure TForm3.ListBox1Click(Sender: TObject);
begin
  filepointer:=listbox1.itemindex;
  form3.visible:=false;
  form4.visible:=true;
end;
```

The **Itemindex** property for the *ListBox* gives the number of the line selected. We will record this as the variable '**filepointer**' so that the correct house record can be loaded and displayed. Add '**filepointer**' to the *Public declarations*:

```
public
    { Public declarations }
    filepointer:integer;
  end;
```

Also add '**unit4**' to the '**uses**' line:

```
     uses
       SysUtils, WinTypes,....StdCtrls,unit4;
```

We now need a procedure to load and display details of the selected property on the **Form 4** screen. Bring **Form 4** to the front. Click on the dotted grid and press ENTER, then click the **Events** tab.

Double-click '**OnActivate**' to produce an event handler, then add the lines:

```
procedure TForm4.FormActivate(Sender: TObject);
begin
  with form1 do
  begin
    assignfile(housefile,'houses.dat');
    reset(housefile);
    seek(housefile,form3.filepointer);
    read(housefile,houserecord);
    closefile(housefile);
  end;
  edit1.text:=form1.houserecord.address;
  edit2.text:=floattostrf
             (form1.houserecord.price,ffFixed,8,0);
  edit3.text:=inttostr(form1.houserecord.bedrooms);
  combobox1.itemindex:=form1.houserecord.housetype;
  combobox2.itemindex:=form1.houserecord.land;
  combobox3.itemindex:=form1.houserecord.location;
end;
```

This begins by opening the house file:

```
      assignfile(housefile,'houses.dat');
      reset(housefile);
```

We then use the SEEK command and the '**filepointer**' variable to load the required record from the file; this will correspond to the address line selected in the '**Houses available**' list:

```
      seek(housefile,form3.filepointer);
      read(housefile,houserecord);
```

The lines:

```
     combobox1.itemindex:=form1.houserecord.housetype;
     combobox2.itemindex:=form1.houserecord.land;
     combobox3.itemindex:=form1.houserecord.location;
```

make use of the code numbers for property type, land and location to select and display the correct text items from the Combo Box lists which we set up earlier.

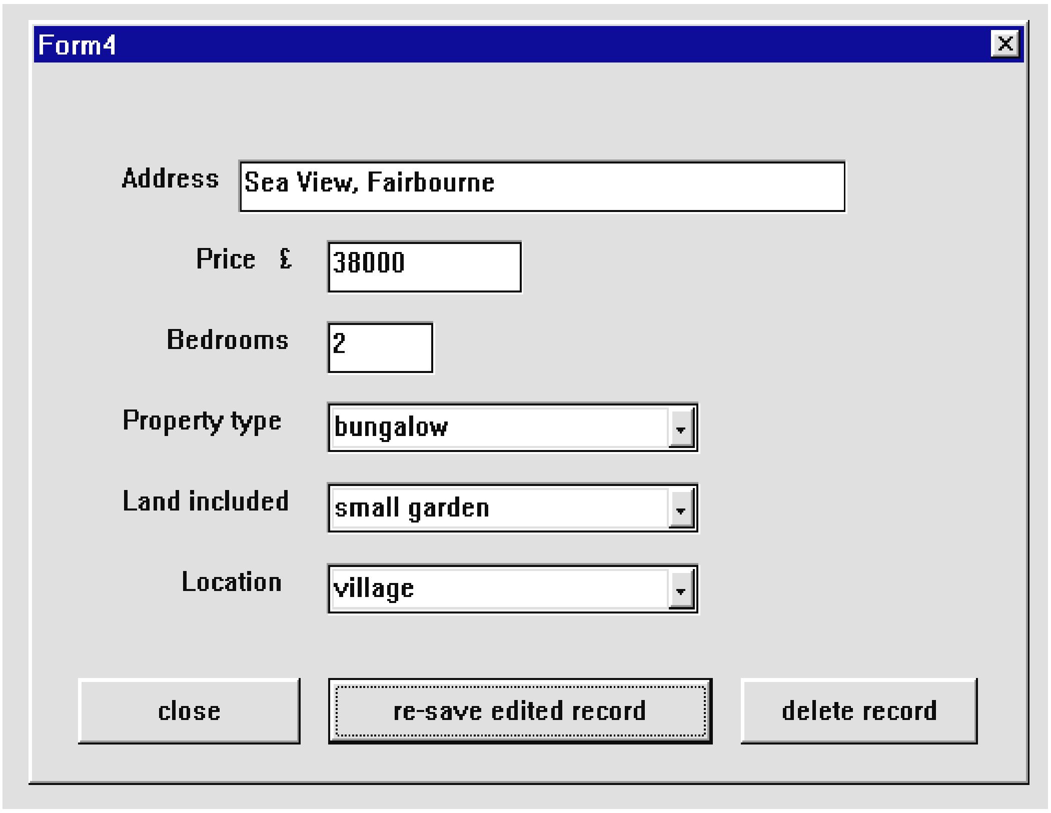
Double-click the '**close**' button of Form4 to create an event handler, then add the lines to close the window and return to the '**Houses available**' list:

```
procedure TForm4.Button1Click(Sender: TObject);
begin
  form4.visible:=false;
  form3.visible:=true;
end;
```

Finally add a '**uses**' command under the implementation heading, so that variables from *Forms 1 and 3* can be accessed:

```
implementation
{$R *.DFM}
uses
  unit1,unit3;
```

Build and run the program. Select the option '**Display house index**'. When the list appears, click on '*Sea View, Fairbourne*' and details should be displayed:



Address: Sea View, Fairbourne
Price £: 38000
Bedrooms: 2
Property type: bungalow
Land included: small garden
Location: village

Use the '**close**' button to return to the '**Houses available**' list, then check that details of the other properties can also be displayed. Finally exit to the Delphi editing screen.

We have made a good start in setting up the estate agent's database. The '**edit**' and '**delete**' options for house records will be completed in the second chapter covering this project.

SUMMARY

In this chapter you have:

- seen a top-down structure diagram for a multi-file database
- set up a menu system using the *Main Menu* component, including a *drop-down* menu
- set up a *record* structure containing *fields* of different data types
- made use of the '**uses**' command to allow one *Form* to access variables belonging to another *Form*
- seen how the '**itemindex**' property of a *List Box* indicates which item has been selected when the mouse is clicked on the list
- used the '**seek**' command to load a particular record from a file
- used a *Combo Box* component to display one a set of text strings