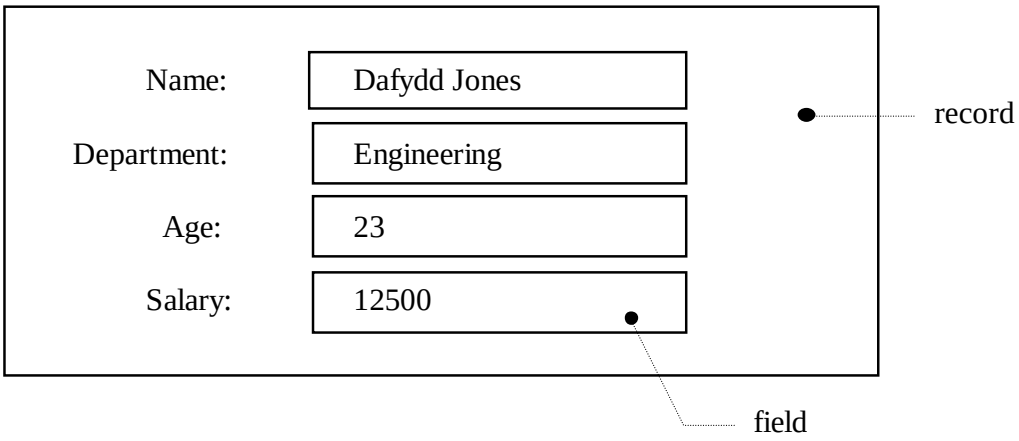
SIX

Saving data on disc

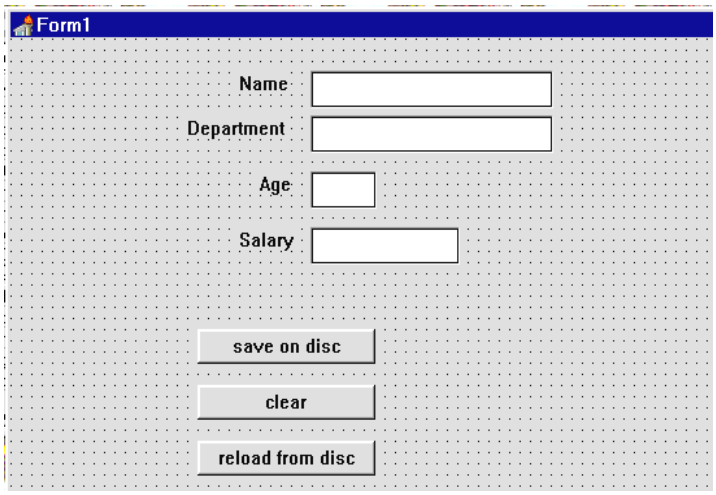
It is often important for computer programs to save data on disc so that the information will not be lost when the machine is switched off. Data on disc can be kept as a permanent record, and reloaded at a future date when needed. As an example we will produce a program to save information about an employee of a company:



The entire set of information relating to the employee (name, department, age and salary) is called a '**record**', whilst an individual data item (e.g. salary) is called a '**field**'.

Make a new sub-directory STAFF in your work area. Open a Delphi project and save this into the sub-directory. Use the Object Inspector to **maximize** the form, and drag the grid to nearly fill the screen.

Place *edit* boxes and *labels* on the form to represent the fields of the record. Also add three *buttons* and give these captions: 'save on disc', 'clear' and 'reload from disc':



The next step is to tell the

computer what fields are to be included in the record which will be stored on disc. This is done beneath the public declarations section. Add the lines:

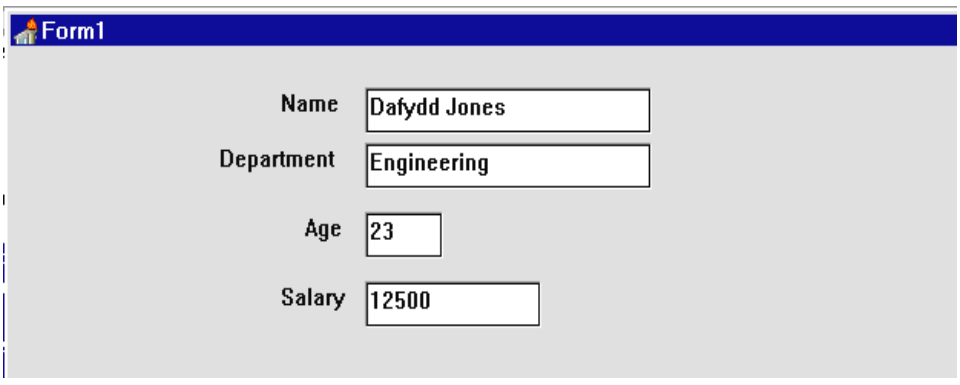
```
  public
     { Public declarations }
   end;

   person=record
     name:string[40];
     department:string[20];
     age:string[3];
     salary:string[8];
   end;

 var
   Form1: TForm1;
   personrecord:person;
```

We have first defined the fields of the record, giving the maximum number of characters which can be typed in - for example, we are allowing the *name* field to contain up to 40 characters. We then give a variable name '**personrecord**' which will be used in the program to refer to the record as data is being entered or stored.

The purpose of the 'clear' button is to blank out the four edit boxes. Double-click this button to create an event handler and add the program lines:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  edit1.text:='';
  edit2.text:='';
  edit3.text:='';
  edit4.text:='';
end;
```

Compile and run the program to check that text can be entered and cleared from the edit boxes.



The next step is to save the record on disc. Produce an event handler procedure for the 'save on disc' button and add the lines shown below:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  personfile:file of person;
begin
  personrecord.name:=edit1.text;
  personrecord.department:=edit2.text;
  personrecord.age:=edit3.text;
  personrecord.salary:=edit4.text;
  assignfile(personfile,'staff.dat');
  rewrite(personfile);
  write(personfile,personrecord);
  closefile(personfile);
end;
```

There is a lot happening here! The lines of program carry out the following tasks:

```
  personfile:file of person;
```

warns the computer that records are to be stored in a file. We will refer to this as 'personfile'.

```
  personrecord.name:=edit1.text;
  personrecord.department:=edit2.text;
  personrecord.age:=edit3.text;
  personrecord.salary:=edit4.text;
```

is a set of lines which copy the entries from the four edit boxes into the fields of the record.

```
  assignfile(personfile,'staff.dat');
```

gives our file of records the name '**staff.dat**' when it is saved on the disc - this is the name which will appear when you list the files in your work directory.

```
  rewrite(personfile);
```

creates a new file on the disc ready to receive the data.

```
  write(personfile,personrecord);
```

stores the record in the empty disc file.

```
  closefile(personfile);
```

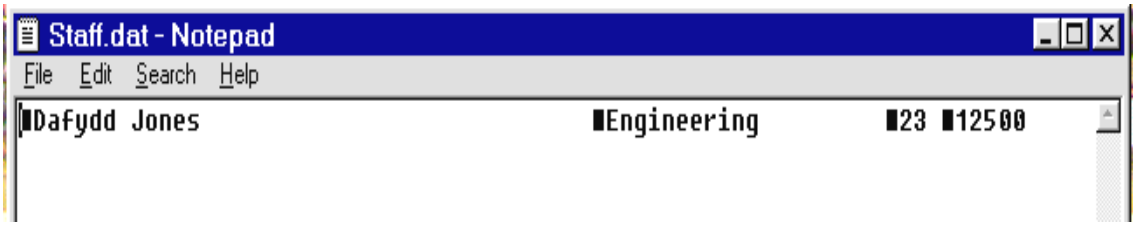
closes the file so that the record we have saved is safe.

Compile and run the program. Enter the test data for Dafydd Jones. Click the 'save on disc' button, then exit from the program and return to the Delphi editing screen.

To check whether the data has been saved, use the Windows **START** button in the bottom left hand corner of the screen to load the **NOTEPAD** utility program. Use the FILE / OPEN option and locate your sub-directory STAFF. Give the file name:

staff.dat

and the saved text should appear:



We can now return to Delphi and set up a procedure to reload the record from disc.

Double-click the 'reload from disc' button to create an event handler and add the lines:

```
procedure TForm1.Button3Click(Sender: TObject);
var
  personfile:file of person;
begin
  assignfile(personfile,'staff.dat');
  reset(personfile);
  read(personfile,personrecord);
  closefile(personfile);
  edit1.text:=personrecord.name;
  edit2.text:=personrecord.department;
  edit3.text:=personrecord.age;
  edit4.text:=personrecord.salary;
end;
```

This is the exact opposite of the saving procedure which you wrote earlier.

```
read(personfile,personrecord);
```

loads in the record from the disc.

```
edit1.text:=personrecord.name;
```

displays the entry for the name field in edit box 1.

Compile and run the program, but this time don't enter the test data from the keyboard. Instead, press the 'load from disc' button and the information you stored earlier should be displayed in the edit boxes.

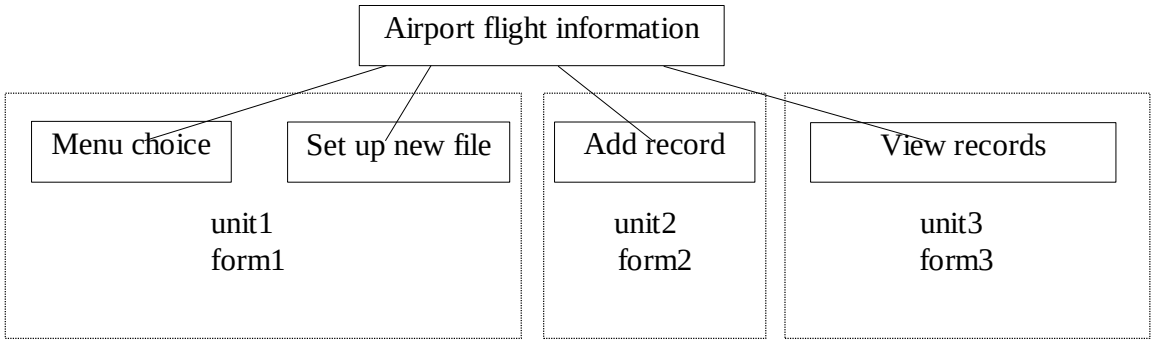
Saving a series of records in a file

In the previous section we successfuly saved and reloaded a single record. In a real system, however, the file is likely to contain many records - one for each employee of the company. We will see how a series of records are stored in our next project:

> An airport requires a computer program to store information about the destination, departure time and airline for each flight scheduled. The program should let the user:
> - set up a new file to hold the records
> - add new records to the file
> - view the records.

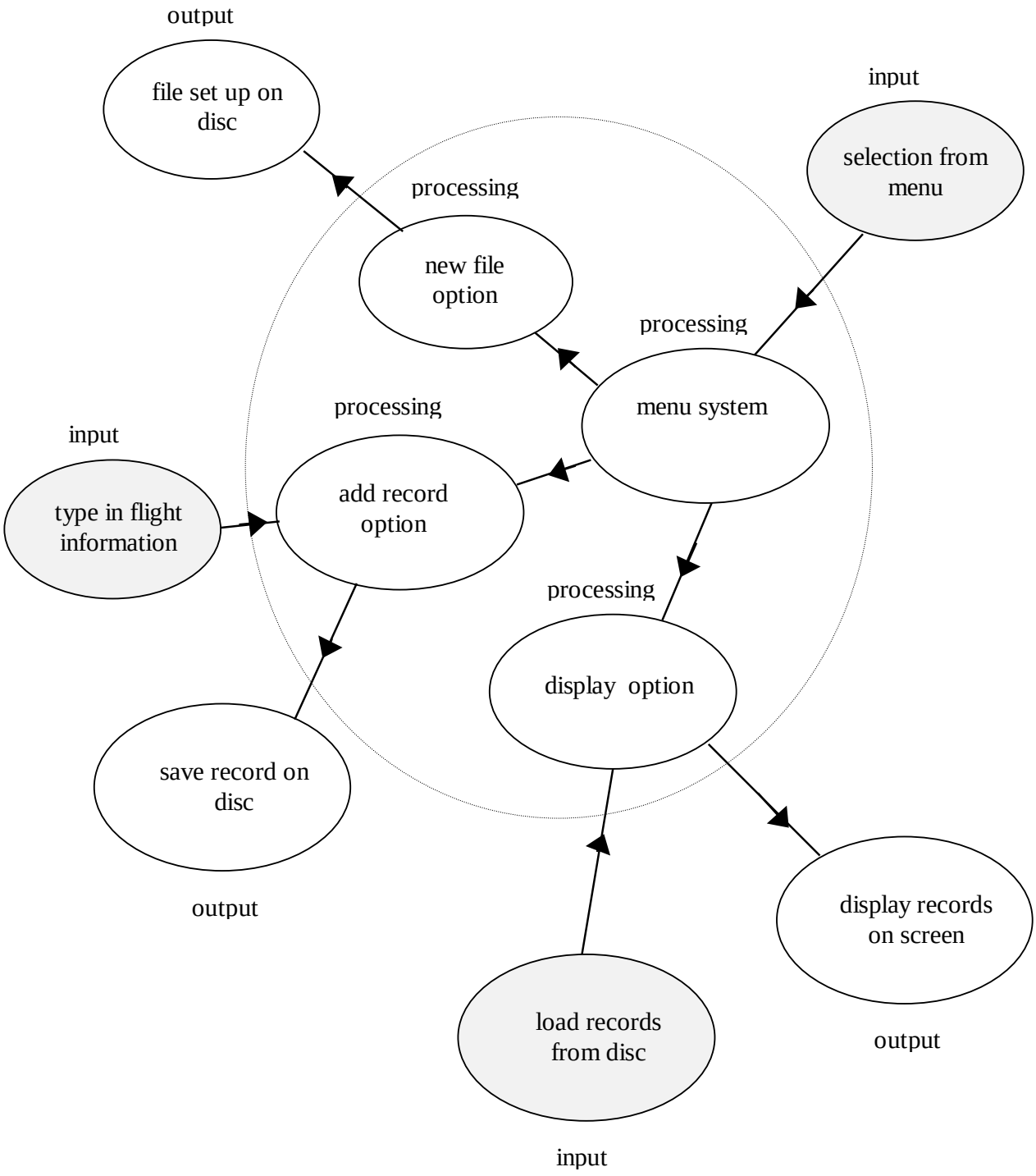
This is quite a complex task, so the first step should be to draw a schematic diagram to clarify exactly what input, processing and output operations will be needed. This is done for you on the next page.

We might now draw a top-down structure diagram to simplify the project into a series of modules:



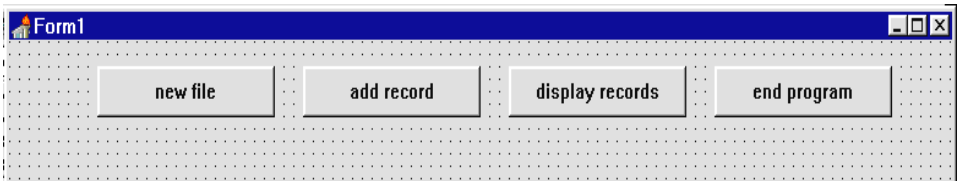
We can arrange for Unit 1 to operate the menu system and setting up of a new file, Unit 2 can deal with adding records, and Unit 3 can handle viewing of records in the file. In this way, the overall problem will be divided into simpler modules which will be easier to program with less risk of errors.

PROGRAM SCHEMATIC


output
file set up on disc
input
selection from menu
processing
new file option
processing
menu system
processing
input
type in flight information
add record option
processing
display option
save record on disc
output
display records on screen
output
load records from disc
input

Set up a new sub-directory AIRPORT in your work area. Open a Delphi project and save it into the sub-directory. Use the Object Inspector to maximize form1, and drag the grid to nearly fill the screen.

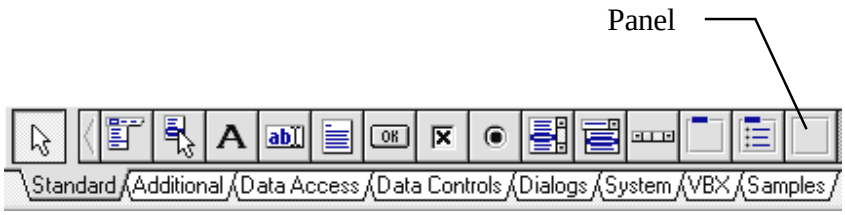
Add four buttons for the program options '*new file*', '*add record*', '*display records*', and '*end program*':



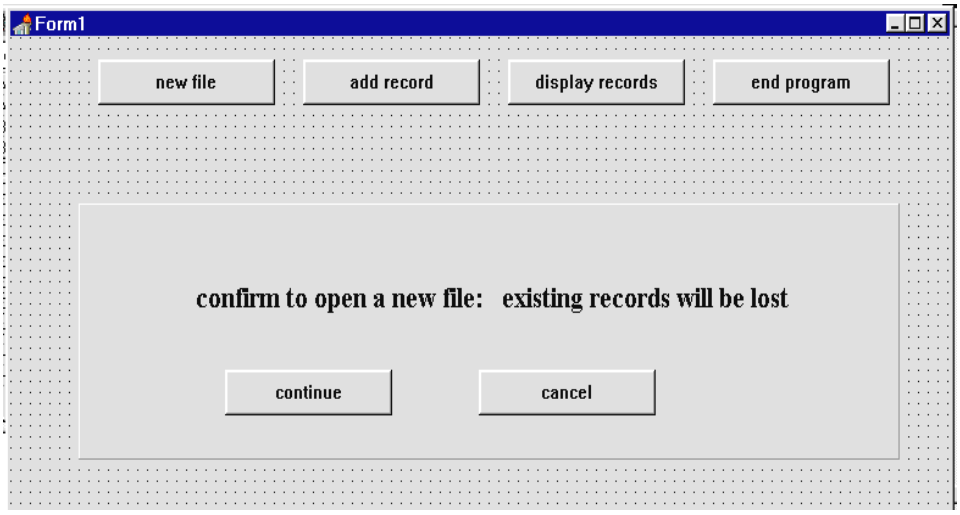
Create an event handler for the '*end program*' button:

```
procedure TForm1.Button4Click(Sender: TObject);
begin
  halt;
end;
```

We can now start work on the '*new file*' option which will also be carried out by Form1. Select the '*panel*' component from the STANDARD menu:



Drag with the mouse to place a panel on the grid. Press ENTER to bring up the Object Inspector and blank out the '**Caption**' property.



Add two buttons to the panel as shown above, and give these the captions '*continue*' and '*cancel*'. Also add a label with the warning message:

confirm to open a new file: existing records will be lost

The *panel* component provides a convenient way of turning on or off a group of buttons or labels. When the panel is closed, all the components on it will also disappear.

Click on the panel and press ENTER to bring up the Object Inspector. Set the '**Visible**' property to '**False**'. Double-click the 'new file' button at the top of the form to create an event handler and add the line:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  panel1.visible:=true;
end;
```

Now double-click the 'cancel' button of the panel and add the event handler line:

```
procedure TForm1.Button6Click(Sender: TObject);
begin
  panel1.visible:=false;
end;
```

Compile and run the program. It should be possible to make the panel and its buttons appear by clicking the '*new file*' button, and disappear by clicking '*cancel*'. Use the '*end program*' button to return to the Delphi editing screen.

Next we will produce the event handler for the '*continue*' button. This is to create the new empty file ready for records to be saved on disc. Add the lines of program:

```
procedure TForm1.Button5Click(Sender: TObject);
var
  flightfile:file of flight;
begin
  assignfile(flightfile,'flights.dat');
  rewrite(flightfile);
  closefile(flightfile);
  panel1.visible:=false;
end;
```

The procedure will set up an empty file in your AIRPORT sub-directory called '**flights.dat**'.

It is necessary to add details of the fields to be included in a flight record. Do this below the '*public declarations*' section:

```
public
    { Public declarations }
  end;

  flight=record
    destination:string[30];
    departtime:string[10];
    airline:string[20];
  end;
```

Compile and run the program. Select '*new file*' and press the '*continue*' button on the panel, then exit from the program. Use the Windows Explorer utility to check that a file '**flights.dat**' has been created - its size should be shown as 0KB.

We can now add the forms for the other modules of the program. Click the 'new form' short cut button and select 'blank form'. A new window for Form2 should appear. Repeat this to create another window for Form3. Use the 'save project' button to save the new units into your AIRPORT sub-directory.

Bring the Form1 grid to the front and create an event handler for the 'add record' button':

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  form2.visible:=true;
end;
```

Create a similar event handler for the 'display records' button:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
  form3.visible:=true;
end;
```

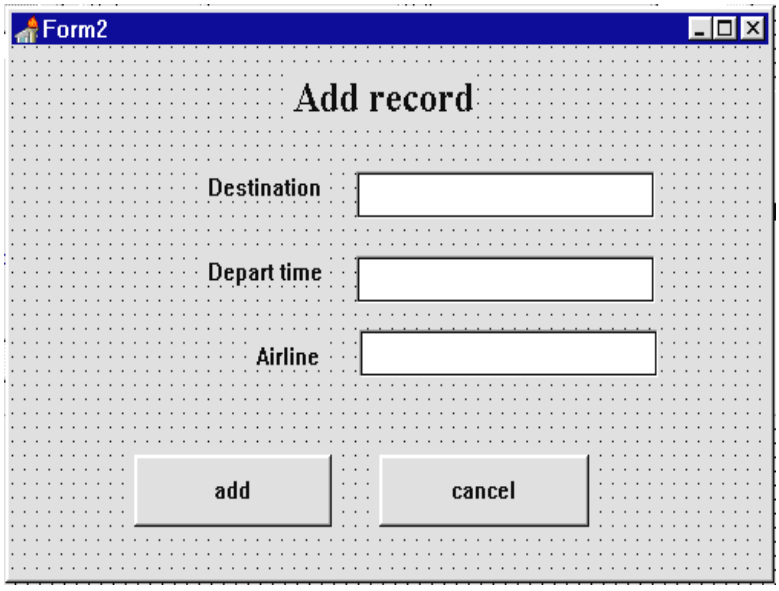
Add the new units to the 'uses' list near the top of the program:

```
uses
  SysUtils,WinTypes,WinProcs,Messages,Classes,Graphics,
Controls,Forms,Dialogs,StdCtrls,ExtCtrls,unit2,unit3;
```

Use the ***project manager*** to select the F**orm2** grid. Press ENTER to bring up the Object Inspector and set the **FormStyle** property to '*StayOnTop*'. Set the **BorderStyle** property to '*Dialog*'.

Repeat this for the **Form3** window, setting the **FormStyle** to '*StayOnTop'* and the **BorderStyle** to *'Dialog'*.

Return to the **Form2** grid and add labels and edit boxes for inputting *destination*, *depart time* and *airline* as shown below. Use a label for the heading '*Add a record*'. Complete the form with two buttons captioned '*add*' and '*cancel*'.



Create an event handler for the '*cancel*' button:

```
procedure TForm2.Button2Click(Sender: TObject);
begin
  form2.visible:=false;
end;
```

Compile the project using the '**Build All**' option and run the program. Check that the '*add record*' button will open the Form2 window, and that this can be closed by the '*cancel*' button. Return to the Delphi editing screen.

The next step is to set up an event handler for the '*add'* button. Double-click to create the procedure then insert the lines of program shown below:

```
procedure TForm2.Button1Click(Sender: TObject);
var
  flightrecord:flight;
  flightfile:file of flight;
begin
  flightrecord.destination:=edit1.text;
  flightrecord.departtime:=edit2.text;
  flightrecord.airline:=edit3.text;
  assignfile(flightfile,'flights.dat');
  reset(flightfile);
  seek(flightfile,filesize(flightfile));
  write(flightfile,flightrecord);
  closefile(flightfile);
  form2.visible:=false;
end;
```

Notice that we begin by telling the computer that we will be using a **flightrecord** in the procedure, and want this to be stored in a **flightfile**:

```
var
  flightrecord:flight;
  flightfile:file of flight;
```

The next lines:

```
flightrecord.destination:=edit1.text;
flightrecord.departtime:=edit2.text;
flightrecord.airline:=edit3.text;
```

transfer the information from the edit boxes to the fields of the **flightrecord**. We then open the **flightfile** with:

```
assignfile(flightfile,'flights.dat');
reset(flightfile);
```

The line:

```
seek(flightfile,filesize(flightfile));
```

moves to the end of the file where the new record will be added - we will discuss how the SEEK command works in more detail later in the course.

The final group of lines:

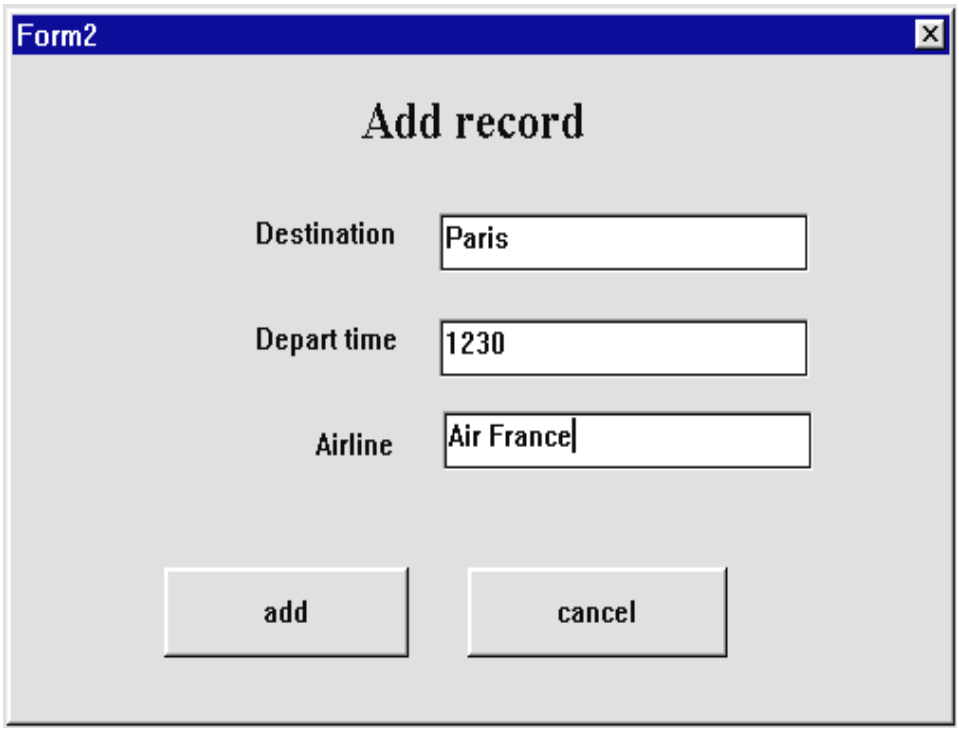
```
write(flightfile,flightrecord);
closefile(flightfile);
form2.visible:=false;
```

add the **flightrecord** to the file, close the file, then close the '*add record*' window.

Below the 'public declarations' section, give the program the necessary information about the fields in a flight record:

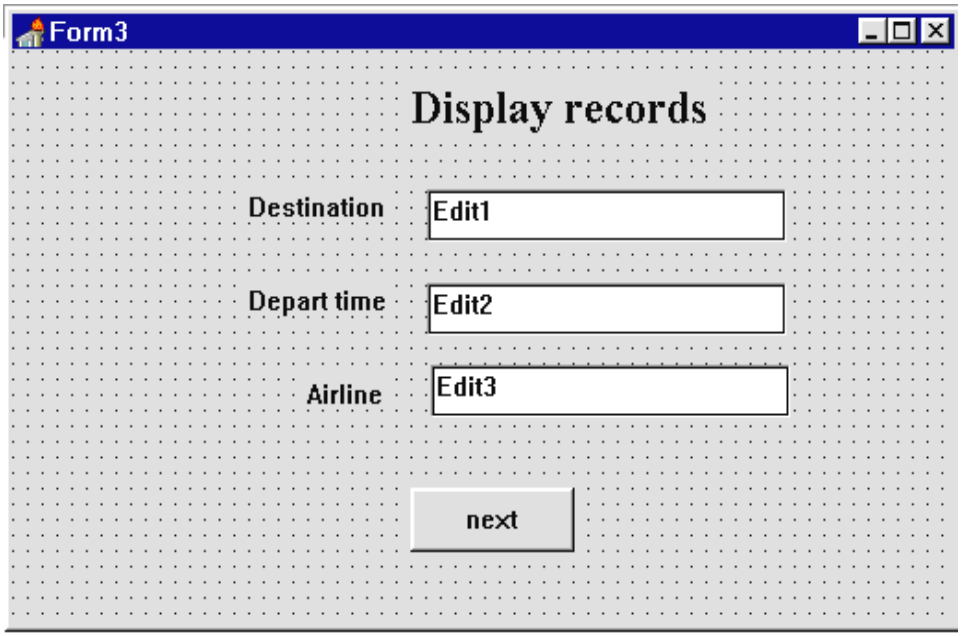
```
public
  { Public declarations }
end;

flight=record
  destination:string[30];
  departtime:string[10];
  airline:string[20];
end;
```

We can now test the input procedure. Compile the program with the '**Build All**' option, then **Run**. First press the '*new file*' button and confirm to create a new file. Go next to the '*add record*' screen and type information about a flight:



Save the record by pressing '*add*', then exit from the program. Use the NOTEPAD utility to check that a file '**flights.dat**' has been created and contains the record you entered (don't worry if there are other random characters present in the file as well as your data - these will be ignored when the record is reloaded by the program).

The next step is to set up edit boxes on Form3 to display the records when they are reloaded from disc. Add captions and a button labelled '*next*':



Click on the form grid then press ENTER to bring up the Object Inspector. Click the '**Events**' tab at the bottom of the window, then double-click the right hand column alongside '**OnActivate**' - this will create an event handler procedure which will run immediately the *'Display records*' window is opened. Add the lines of program:

```
procedure TForm3.FormActivate(Sender: TObject);
begin
  assignfile(flightfile,'flights.dat');
  reset(flightfile);
  if not eof(flightfile) then
  begin
    read(flightfile,flightrecord);
    edit1.text:=flightrecord.destination;
    edit2.text:=flightrecord.departtime;
    edit3.text:=flightrecord.airline;
  end;
end;
```

The procedure begins by opening the flightfile with the commands:

```
assignfile(flightfile,'flights.dat');
reset(flightfile);
```

The line:

if not eof(flightfile) . . .

checks to see whether there are any records in the file. The letters '**eof**' stand for '*end of file*'. If the end of the file has not been reached then the computer will load a record:

read(flightfile,flightrecord);

The data is then transferred to the edit boxes and displayed:

edit1.text:=flightrecord.destination;
edit2.text:=flightrecord.departtime;
edit3.text:=flightrecord.airline;

Add the field information for a flight record below the '*public declarations*' section, as you did with Form2. **Flightrecord** and **flightfile** are also added at this point:

```
    { Public declarations }
  end;

  flight=record
    destination:string[30];
    departtime:string[10];
    airline:string[20];
  end;

var
  Form3: TForm3;
  flightrecord:flight;
  flightfile:file of flight;
```

We can finish off Form3 by adding an event handler for the '*next*' button:

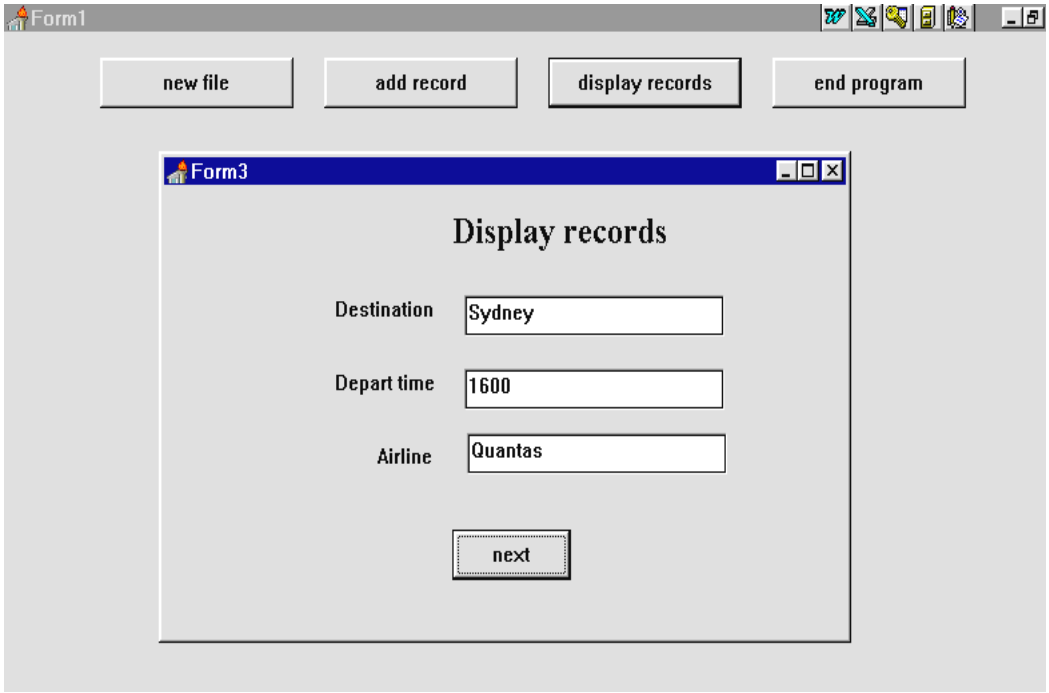
```
procedure TForm3.Button1Click(Sender: TObject);
begin
 if not eof(flightfile) then
  begin
    read(flightfile,flightrecord);
    edit1.text:=flightrecord.destination;
    edit2.text:=flightrecord.departtime;
    edit3.text:=flightrecord.airline;
  end
  else
  begin
    closefile(flightfile);
    form3.visible:=false;
  end;
end;
```

This procedure will begin by checking whether the end of the file has been reached yet. If not, then a flight record will be loaded and displayed in the edit boxes.

If it is found that there are no more records to display, **flightfile** will be closed and the '*Display records*' window will close down.

Compile the program using '**Build All**' then **Run**. Go directly to the '*display records*' option and the details of the Paris flight which you entered earlier should be displayed. Pressing '*next*' will close the display window.

Press '*add record*' and enter details of another flight. Choose '*display records*' again and the program should show the two records now in the file:



Further records can be added, but you will find that it is necessary to delete the previous entries from the edit boxes in the '*Add record*' window each time. To avoid this we will add a procedure to blank out the form automatically.

Exit from the program and return to the Delphi editing screen. Use the Project Manager to bring Form2 to the front. Click on the grid and press

ENTER to bring up the Object Inspector. Click the '**Events**' tab at the bottom, then double-click alongside '**On Activate**'. Add lines of program to the event handler which appears:

```
procedure TForm2.FormActivate(Sender: TObject);
begin
  edit1.text:='';
  edit2.text:='';
  edit3.text:='';
end;
```

Build and **Run** the program. Add several more flight records and check that they are saved and reloaded correctly.

This completes the introduction to saving data on disc. In a later section of the course you will be setting up a more complete database program which will allow existing records to be edited and deleted.

SUMMARY

In this chapter you have:

- set up record structures containing a number of data fields
- used '**assignfile**' to specify the name of a data file on disc
- used '**rewrite**' to create a new empty file
- used '**reset**' to open an existing file
- used '**write**' to save a record into a file
- used '**read**' to load a record from a file
- used the '**seek**' command to add records to the end of an existing file
- used '**closefile**' to close a file so that data is secure
- transferred data between the fields of a record and edit boxes on screen
- used the '*panel*' component to show and hide groups of buttons and labels