Millionaire investment

1. The task

You have been commissioned to write a Delphi program to assist winners on a TV quiz show to manage their prize money. They may wish to invest it for a number of months, and they may wish to make purchases.
In order to find the result of investing the money, the user enters a number of months in the middle edit box. She/he then clicks the 'Press to invest' button. The program should then work out monthly interest and update the topmost edit box based on the following interest rates
1% per month on capital over £10,000
0.9% per month on capital over £1,000 (but less than or equal to £10,000)
0.8% per month on smaller capital sums.
The funds may reach a higher interest rate over a period of investment. Include these six figures as constants.

For each month, the program should multiply capital by the appropriate interest rate and add the interest earned to the capital. Thus the capital for the next month is greater.

The user may wish buy an item. To see the effect on the capital of doing so the user enters price in the bottom edit box and then clicks the 'Press to spend' button. The program should check that funds are available, and display a message box if not. If funds are available the top edit box should be updated.

The user may wish to invest for a number of months, then make purchases, then invest again. However he/she cannot make purchases during an investment period.
For instance, suppose funds start at £16,000, then the user buys first a PC for £5,000 then Hi Fi for £1,050. She then invests the remainder for 3 months, after that period the fund will stand at £10,241.34.

Use a spreadsheet to verify a different scenario of your choice with at least two separate investment periods.

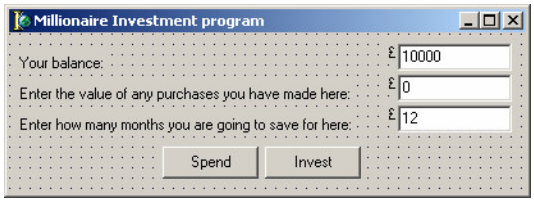
2. Design and Development

Initial design:

In my design there need to be three text boxes and two buttons in accordance with the problem outline. I created this user interface as my initial design.

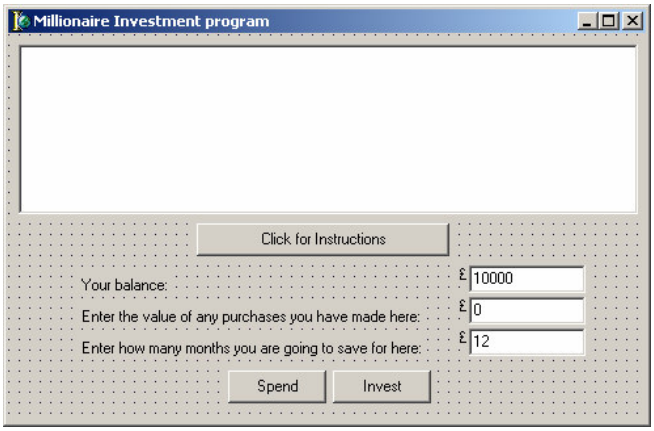
Developing the design:

It is not clear to the user what figures must be entered in which boxes, so to my design I added two labels for each of the text boxes, one instructing the user on what to enter and one simply a '£' to point out to the user that the figure is a monetary value.



Final design:

In my final design I added a memo box and instructions button. This improvement makes the program I have created much easier for the user to understand. I chose to re-order the text box into which the user enters the value of purchases and the months for investment buttons, as it is likely that purchases will be made before investment, as in the example at the end of the problem description.



Pseudo-code:

For the "Invest" button:

```
Money ← Money_textbox
Months ← Months_textbox

LOOP FOR 1 TO Months DO
   IF Money>10000 THEN
      Money ← Money * 1.01
   IF Money<=10000 AND Money>1000 THEN
      Money ← Money * 1.009
   IF Money<=1000 AND Money>0 THEN
      Money ← Money * 1.008
END LOOP

   IF Money<0 THEN
      DISPLAY_ERROR "Invalid data entry"

PRINT Money_textbox ← Money
```

For the "Spend" button:

```
Money ← Money_textbox
Purchases ← Purchases_textbox

RemainingMoney ← Money – Purchases

   IF RemainingMoney>=0 THEN
      PRINT Money_textbox ← RemainingMoney
   ELSE
      DISPLAY_ERROR "You do not have enough funds to make this purchase"
```

For the "Instructions" button:

```
  PRINT "Millionaire Investment programme"
  PRINT "--------------------------------"
  PRINT "Enter your winnings in the balance box. Enter any purchases you make in the second edit box and
click the 'Spend' button. The remaining sum is re-entered into your balance box. Enter the number of months
you wish to invest the remaining sum for and click the 'Invest' button. The sum you will gain by interest is
calculated and re-entered into your balance box."
```

Here is a data dictionary for this project:

Identifier	Type	Formatting	Reasoning
Money	Real	2 decimal places	Two figures for pence, no limit on numerical value of pounds.
Purchases	Real	<=Money, 2 decimal places	Two figures for pence, no limit on numerical value of pounds. Less than or equal to balance or puchase is not possible.
RemainingMoney	Real	2 decimal places	Two figures for pence, no limit on numerical value of pounds.
Months	Cardinal		Must be positive whole number, as interest is paid at monthly time periods.
i	Cardinal		Must be positive whole number (number of iterations)

3. Problems

During the process of creating the program I encountered a number of errors.

Initially I had written the error message within the loop. I had originally considered the if statement for Money and the values outlined in the problem. If these conditions were not true, (ie: the money is negative) this cannot be invested and so an error message is to be generated.

I then enclosed this if statement with a loop, as if after an interest payment the balance increases to over a boundary (£1000, £10000) then on the next iteration of the loop the interest rate will have increased to the higher rate.

However, if the value entered was negative and invested for, say, 12 months then the error message would have been displayed yet after clicking "OK" it would appear a further 11 times as each iteration of the loop would successively generate it.

Therefore I moved the message code to outside of the loop and so resolved my problem.

During testing I also noticed that I had omitted the "=" part of the If (Money<=1000) expression, noticed this and corrected it.

4. Unit Listing

Here is the Pascal source code copied from Delphi.

```
unit UMillionaire;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TfrmMain = class(TForm)
    txtMoney: TEdit;
    txtMonths: TEdit;
    txtPurchases: TEdit;
    btnInvest: TButton;
    lblMoney: TLabel;
    lblMonths: TLabel;
    lblPurchases: TLabel;
    btnSpend: TButton;
    mmoInstructions: TMemo;
    btnInstructions: TButton;
    lblPounds1: TLabel;
    lblPounds2: TLabel;
    lblPounds3: TLabel;
    procedure btnInvestClick(Sender: TObject);
    procedure btnSpendClick(Sender: TObject);
    procedure btnInstructionsClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmMain: TfrmMain;

implementation

{$R *.dfm}

//
// *** INVEST BUTTON ***
//
procedure TfrmMain.btnInvestClick(Sender: TObject);
var
//
// Read values of Money and Months from text boxes
//
Money: Real;
Months, i: cardinal;
begin
Money:= StrToFloat(txtMoney.Text);
Months:= StrToInt(txtMonths.Text);
```

```
//
// Loop repeated for the number of months
//
 for i:= 1 to Months do begin
//
// If statements executed each time the loop is repeated. Money is incremented by a certain percentage
//
  if (Money>10000) then
    Money:= (Money*1.01)
  else
  if (Money<=10000) and (Money>1000) then
    Money:= (Money*1.009)
  else
  if (Money<=1000) and (Money>0) then
    Money:= (Money*1.008);
 end;
//
// If a negative value is entered for Money then a message is displayed...
//
 if Money<0 then
  ShowMessage('Value entered for money not valid.')
//
// ...otherwise the new value for the balance is written to the Money edit box.
//
  else
  txtMoney.Text:= FloatToStrF(Money,ffFixed,15,2);
end;

//
// *** SPEND BUTTON ***
//
procedure TfrmMain.btnSpendClick(Sender: TObject);
var
Money, Purchases, RemainingMoney: Real;
begin
//
// Read values from text boxes Money and Purchases
//
Money:= StrToFloat(txtMoney.Text);
Purchases:= StrToFloat(txtPurchases.Text);
//
// Calculate the remaining funds
//
RemainingMoney:= Money - Purchases;
//
// Generate error if not enough money to make purchases,
// Else show new balance
//
 if (RemainingMoney>=0) then
  txtMoney.Text:= FloatToStrF(RemainingMoney,ffFixed,15,2)
 else
  ShowMessage('You do not have enough funds to make this purchase.');
end;

//
// *** INSTRUCTIONS BUTTON ***
//
procedure TfrmMain.btnInstructionsClick(Sender: TObject);
begin
//
// Print instructions as lines in a memo field.
//
  mmoInstructions.Lines.Add('Millionaire Investment programme');
  mmoInstructions.Lines.Add('--------------------------------');
  mmoInstructions.Lines.Add('Enter your winnings in the balance box.');
  mmoInstructions.Lines.Add('Enter any purchases you make in the second edit box and click the "Spend"
button.');
  mmoInstructions.Lines.Add('The remaining sum is re-entered into your balance box.');
  mmoInstructions.Lines.Add('Enter the number of months you wish to invest the remaining sum for and click
the "Invest" button.');
```

```
  mmoInstructions.Lines.Add('The sum you will gain by interest is calculated and re-entered into your
balance box.');
end;

end.
```

5. Testing

Instructions button

Test	Why testing?	Expected outcome	Actual outcome
Click instructions button	Check instructions are correctly displayed	The following text: "Millionaire Investment programme ---------- Enter your winnings in the balance box. Enter any purchases you make in the second edit box and click the "Spend" button. The remaining sum is re-entered into your balance box. Enter the number of months you wish to invest the remaining sum for and click the "Invest" button. The sum you will gain by interest is calculated and re-entered into your balance box." is displayed in memo box, mmoInstructions.	As expected.

Spend button

Test	Why testing?	Expected outcome	Actual outcome
Balance: £10,000.00 Purchases: £9,999.99	Checking differences between balance and purchases calculated correctly. Checking the boundaries, before on and after updated balance is 0. This implies all other values will be calculated correctly.	Balance updated to £0.01	As expected
Balance: £10,000.00 Purchases: £10,000.00		Balance updated to £0.00	As expected
Balance: £10,000.00 Purchases £10,000.01		Error message "You do not have enough funds to make this purchase" generated	As expected

Invest button

Test	Why testing?	Expected outcome	Actual outcome
Months: 0 Balances: £10,000.01, £10,000.00, £9999.99, £1000.01, £1000.00, £999.99	Checking 0 months is recognised as no time so no interest is earnt. Checking boundaries.	Balance remains at original balance as no interest is earnt	As expected
Months: 1 Balances: £10,000.01, £10,000.00, £9999.99, £1000.01, £1000.00, £999.99	Checking interest correctly applied over period of one month. Checking boundaries.	Balance has appropriate interest added to it, as described in the problem.	As expected apart from for £1,000.00*

* For £1,000.00 the balance remained the same. An inspection of the code revealed that I had omitted the "=" sign in the expression 'if (Money<=1000).' Once I had added the "=" I re-tested and the above tests all still worked.

Test	Why testing?	Expected outcome	Actual outcome
Months: 2 Balances: £10,000.01, £10,000.00, £9999.99, £1000.01, £1000.00, £999.99	Checking interest correctly applied over period of two months. Checking boundaries as for some amounts the second interest rate will be different to the first as the amount has passed a boundary.	Balance has appropriate interest added to it, as described in the problem	As expected.
Months: -1 Balances: £10,000.01, £10,000.00, £9999.99, £1000.01, £1000.00, £999.99	Check that an invalid month figure is not accepted as it is not a cardinal.	Error "Floating point overflow"	As expected
Months: 1 Balance £-£0.01	Checking negative value not acceptable investment	Error "Value entered for money not valid"	As expected

I then tested my program further, as the problem requires me to, by choosing two example situations and verifying the results.

1. A contestant wins £50,000 and makes purchases including a car and cooker which total £18,050. For the next three years the contestant invests the remaining sum.

   To enter the data into the program the 3 years must be converted into months, 3 * 12 = 36.

   The program gives the answer of £68891.52 after the investment period.

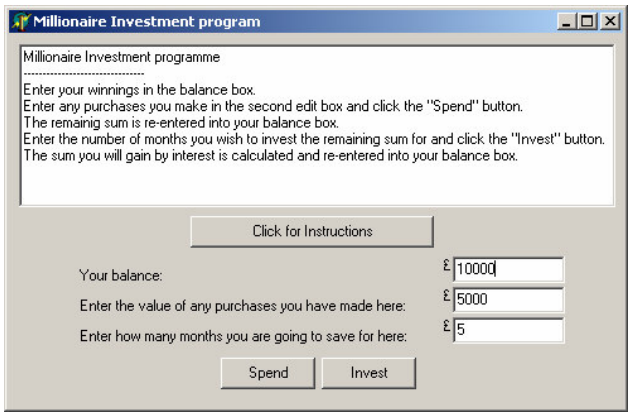
   A spreadsheet verifies this, as 50000 - 18050 = 48150, $48150 * 1.01^{36} = 68891.5169…$

2. A contestant wins £8,000 and spends £4,525 on a new Kitchen. She invests the remainder for a year.

   One year is 12 months, and the program gives and answer of £3869.45

   A spreadsheet verifies this, as $8000 – 4525 = 3475$, $3475 * 1.009^{12} = 3869.4461…$

6. Sample run



7. Appraisal

I have met the requirements of this task, by creating a program that calculates the interest that could be earnt on potential investments, and deal with purchases. I have used a small range of components to create this program which I now feel confident with, have used a loop and 'if' statements which are programming constructs I also now feel confident with.

What has been relatively new to me in this project is the manipulation of real numbers and formatting them to display in a text box. Deciding to format using ffFixed and not ffCurrency for example, as ffCurrency will insert a £ sign in the text box, which makes the entry an invalid real number.

I extended the brief by adding validation routines, instructions and labels to my program. I could further extend it by adding a button or menu option to quit the program and allowing periods of saving between purchases, perhaps printing the output to a memo box in a receipt like format.

I feel that the style with which I code has improved, the identifier names I have chosen are more sensible and the block formatting of the code is clear. The code has been reasonably neatly written and uses only a few variables. This has been helped by the use of pseudo code to plan my code before writing in Delphi.

Section	Contents	Marks	Total
Design	Data and program structures	2	4
10 marks	Pseudo code or flowchart	3	3
	User Interface	3	3
Coding	Functionality	7	8
20 marks	Formatting, annotation and comments	5	5
	Clear relationship to design	2	2
	Bonus marks for extending the brief	2	5
Testing and Evaluation	Test data and evidence	7	7
10 marks	Evaluation	3	3
Total		34	40
	Divided by 2	17	20